PERANCANGAN DAN IMPLEMENTASI MODUL OTENTIKASI MENGGUNAKAN RANDOMISASI *PASSWORD* BERDASARKAN *LOOKUP TABLE*

TUGAS AKHIR

Diajukan sebagai persyaratan mendapatkan gelar Sarjana

pada Jurusan Teknik Informatika

UIN SUSKA RIAU

M. AFFANDES

10551001474

FAKULTAS SAINS DAN TEKNOLOGI

JURUSAN TEKNIK INFORMATIKA

UIN SUSKA RIAU

2010

PERANCANGAN DAN IMPLEMENTASI MODUL OTENTIKASI MENGGUNAKAN RANDOMISASI *PASSWORD* BERDASARKAN *LOOKUP TABLE*

M. AFFANDES

10551001474

Tanggal Sidang : 1 Februari 2010

Periode Wisuda : Februari 2010

Jurusan Teknik Informatika

Fakultas Sains dan Teknologi

Universitas Islam Negeri Sultan Syarif Kasim Riau

ABSTRAK

Menu otentikasi (*authentication*) merupakan menu yang paling sering ditemukan pada saat mengunjungi suatu situs *web*. Menu otentikasi berfungsi untuk memastikan seorang pengguna yang masuk pada suatu situs *web* adalah pemilik data yang sebenarnya. Selain itu, otentikasi ini juga berguna sebagai tanda pengenal seseorang pada suatu situs *web*.

Pada dunia internet juga terjadi pencurian identitas. Sehingga 'pencuri' dapat mengakses informasi orang lain. Hal ini akan sangat berbahaya apabila terjadi terhadap data-data penting yang harus diproteksi, sehingga diperlukan cara yang benar-benar mampu mencegah tindakan pencurian atau setidaknya mencegah pemanfaatan identitas digital tersebut. Untuk itu dirancang sebuah modul otentikasi yang mampu melakukan hal tersebut yang disebut *modAuth*.

modAuth menggunakan sebuah *lookup table* yang diberi nama *mCode* dan diberikan kepada pengguna. Masing-masing pengguna memiliki *mCode* yang berbeda-beda. Pada saat otentikasi, pengguna akan diminta memasukkan kode-kode yang terdapat pada *mCode*-nya secara acak. Apabila kode-kode tersebut berhasil diperoleh menggunakan aplikasi tertentu, tetap tidak akan bisa digunakan. Karena pada saat otentikasi berikutnya, *modAuth* akan meminta kode-kode yang berbeda.

Kata Kunci : ***Keylogger, Login, mCode,* Menu Otentikasi, *modAuth, Password, Sniffer, Username*.**

PLANNING AND IMPLEMENTATION OF AUTHENTICATION MODULE USE PASSWORD RANDOMIZATION BASED ON LOOKUP TABLE

M. AFFANDES

10551001474

Date of Final Exam : February 1st 2010

Graduation Ceremony Period : February 2010

Informatics Engineering Departement

Faculty of Sciences and Technology

State Islamic University of Sultan Syarif Kasim Riau

ABSTRACT

Authentication menu is the very often menu that has visited while surfing a website. The function of authentication menu is to ensure an user that log into a website is the real owner. Authentication also as an identity on a website.

In the internet also happened stealing against identity information. It will make 'the thief' enable to access other people information. It very dangerous if happened to important data which have to protect, as a result that need some methods which really can prevent stealing or at least can prevent exploiting the digital identity. Then design an authentication module which can perform that which called modAuth.

modAuth use a lookup table which called mCode and give to user. Each user have different mCode. While authentication, user will asked to type codes randomly based on user's mCode. If codes got successfully with certain application, codes will not work. Because at next authentication, modAuth will ask different codes.

Keywords : ***Keylogger, Login, mCode, Authentication Menu, modAuth, Password, Sniffer, Username.***

DAFTAR ISI

Halaman

LEMBAR PERSETUJUAN ii
LEMBAR PENGESAHAN iii
LEMBAR HAK ATAS KEKAYAAN INTELEKTUAL iv
LEMBAR PERNYATAAN v
LEMBAR PERSEMBAHAN vi
ABSTRAK vii
ABSTRACT viii
KATA PENGANTAR ix
DAFTAR ISI xi
DAFTAR GAMBAR xiv
DAFTAR TABEL xv
DAFTAR LAMPIRAN xvi

BAB I PENDAHULUAN
1.1 Latar Belakang I-1
1.2 Rumusan Masalah I-3
1.3 Batasan Penelitian I-3
1.4 Tujuan Penelitian I-4
1.5 Sistematika Penulisan I-4
BAB II LANDASAN TEORI
2.1 *Keylogger* II-1
2.1.1 Pengertian *Keylogger* II-1
2.1.2 Cara Kerja *Keylogger* II-1
2.1.3 Mencegah *Keylogger* II-2
2.2 *Sniffer* II-2
2.2.1 Pengertian *Sniffer* II-2
2.2.2 Cara Kerja *Sniffer* II-3
2.2.3 Mencegah *Sniffer* II-4
2.3 *Look Up Table* II-5

2.4 PHP II-7
2.5 Analisa dan Perancangan Berorientasi Objek II-8
2.5.1 *Unified Modelling Language* (UML) II-9
2.5.2 *Use Case Diagram* II-9
2.5.3 *Class Diagram* II-9
2.5.4 *Behavior Diagram* II-10
2.5.5 *Implementation Diagram* II-11
2.6 *Rational Unified Process* (RUP) II-11
2.6.1 Pengertian RUP II-12
2.6.2 Fase RUP II-13
2.7 *Message Diggest* (MD5) II-17
2.7.1 Sekilas Tentang MD5 ..... II-17
2.7.2 Algoritma MD5 ..... II-18
BAB III METODOLOGI PENELITIAN
3.1 Tahapan Penelitian III-1
3.1.1 Pengumpulan Materi III-1
3.1.2 Analisa dan Perancangan Sistem III-2
3.1.3 Implementasi III-2
3.1.4 Pengujian Sistem III-3
3.2 Waktu Penelitian III-3
3.3 Tahapan RUP III-4
3.3.1 Fase *Inception* III-4
3.3.2 Fase *Elaboration* III-4
3.3.3 Fase *Construction* III-4
3.3.4 Fase *Transition* III-5
BAB IV ANALISA DAN PERANCANGAN
4.1 Deskripsi Umum IV-1
4.2 Analisa Modul Lama IV-2
4.2.1 Model Kerja Modul Lama IV-3
4.2.2 Kelemahan Modul Lama IV-4
4.3 Analisa Modul Baru IV-5

4.3.1 Analisa Kebutuhan IV-5
4.3.2 Analisa Fungsional Modul IV-6
4.4 Perancangan Modul IV-12
4.4.1 *Class Diagram* IV-12
4.4.2 *Component Diagram* IV-13
4.4.3 *Statechart Diagram* IV-14
4.4.4 *Deployment Diagram* IV-15
4.4.5 Rancangan *Database* IV-16
4.4.6 Perancangan *Interface* IV-22
BAB V IMPLEMENTASI DAN PENGUJIAN
5.1 Implementasi Modul V-1
5.1.1 Lingkungan Implementasi V-1
5.1.2 Batasan Implementasi V-2
5.1.3 Hasil Implementasi V-3
5.2 Pengujian Modul V-5
5.2.1 Pengujian Fungsional V-6
5.2.2 Pengujian Non-Fungsional V-8
5.2.3 Kesimpulan Pengujian V-13
BAB VI PENUTUP
6.1 Kesimpulan VI-1
6.2 Saran VI-1

DAFTAR PUSTAKA
LAMPIRAN
DAFTAR RIWAYAT HIDUP

BAB I

PENDAHULUAN

1.1. Latar Belakang

Menu otentikasi (*authentication*) merupakan menu yang paling sering kita temukan pada saat mengunjungi suatu situs *web*. Menu ini biasanya diletakkan pada halaman awal suatu situs *web*. Menu otentikasi berfungsi untuk memastikan seorang pengguna yang masuk pada suatu situs *web* adalah pemilik data yang sebenarnya. Ini penting diterapkan untuk menjaga kerahasiaan informasi yang berhubungan dengan orang yang dimaksud seperti *e-mail*, foto-foto yang bersifat pribadi, informasi rekening bank, atau informasi lainnya yang bersifat pribadi. Selain itu, otentikasi ini juga berguna sebagai tanda pengenal seseorang pada suatu situs *web*.

Sama halnya pada kehidupan nyata, tanda pengenal tersebut juga disebut dengan Kartu Tanda Penduduk (KTP) untuk di Indonesia. Dengan adanya kartu identitas ini seseorang dikenal secara sah menurut undang-undang atau hukum yang berlaku. Namun di balik itu semua terdapat kecurangan-kecurangan atau pelanggaran-pelanggaran yang terjadi berkenaan dengan sah atau tidak sahnya pemilik kartu tersebut. Dalam kehidupan nyata, KTP hanya sebuah kartu yang siapa saja punya kemungkinan untuk melakukan pemalsuan terhadap KTP tersebut. Tujuannya melakukan pemalsuan kartu identitas ini bermacam-macam, antara lain untuk mendapatkan keuntungan satu orang atau satu kelompok, dan berbagai macam tujuan lain yang ingin dicapai.

Di dunia digital khususnya dunia internet, kartu identitas tersebut dapat berupa suatu rangkaian huruf dan atau angka yang berbeda-beda untuk setiap orang. Seiring berkembangnya teknologi saat ini, kartu identitas tersebut juga dapat berupa sidik jari seseorang. Dengan teknologi ini, seseorang cukup menempelkan jarinya pada *fingerprint scanner* sebagai cara melakukan otentikasi. Selain sidik jari, telah dikembangkan juga metode otentikasi menggunakan kornea mata, yaitu dengan melakukan *scanning* pada salah satu mata seseorang untuk melakukan otentikasi.

Namun, jika di situs *web* diterapkan model otentikasi menggunakan sidik jari atau kornea mata, maka akan mempersulit seseorang untuk mengakses informasinya. Ini dikarenakan tidak semua pengguna internet memiliki *fingerprint scanner* (pemindai sidik jari) atau *cornea scanner* (pemindai kornea mata). Faktor harga dan distribusi alat juga berpengaruh terhadap jumlah orang yang menggunakannya. Itulah sebabnya model otentikasi berupa rangkaian huruf dan atau angka ini masih populer saat ini. Selain praktis dan cepat, model ini juga tidak memerlukan biaya untuk membeli alat pendukung. Karena alat pendukung hanya berupa *keyboard* atau *mouse*.

Jika dalam dunia nyata terjadi pemalsuan KTP, pada dunia internet justru terjadi pencurian kartu identitas. Sehingga ‘pencuri’ dapat mengakses informasi orang yang kartu identitasnya dicuri atau melakukan sesuatu atas nama korban. Metode pencurian kartu identitas ini sangat banyak, antara lain yang paling populer dan tergolong mudah dilakukan adalah *phising, sniffing, keylogging*, dan lain-lain. *Tools* yang digunakan untuk melakukan cara di atas banyak beredar

secara bebas, sehingga semua orang punya kemungkinan untuk melakukan pencurian kartu identitas digital ini. Hal ini akan sangat berbahaya apabila terjadi terhadap data-data penting yang harus diproteksi, sehingga diperlukan cara yang benar-benar mampu mencegah tindakan pencurian atau setidaknya mencegah pemanfaatan identitas digital tersebut.

Telah banyak cara diterapkan untuk mencegah terjadinya pencurian atau pemanfaatan kartu identitas ini seperti *data encryption* (enkripsi data) dan *virtual keyboard*. Namun cara tersebut hanya mencegah salah satu atau beberapa cara yang tersebut di atas.

Untuk itu perlu dirancang suatu modul otentikasi yang dapat mencegah dari pemanfaatan identitas digital, khususnya untuk melindungi data yang sangat penting. Itulah sebabnya penelitian ini diberi judul "**Perancangan dan Implementasi Modul Otentikasi Menggunakan Randomisasi *Password* Berdasarkan *Lookup Table***".

1.2. Rumusan Masalah

Dari latar belakang yang telah dipaparkan sebelumnya dapat diambil suatu perumusan masalah bahwa perancangan menu *login* yang menggunakan rangkaian huruf dan angka tidak sepenuhnya aman dari pencurian khususnya terhadap aplikasi *keylogger* dan *sniffer*.

1.3. Batasan Penelitian

Untuk menghindari kesalahan persepsi dalam penulisan, berikut merupakan batasan permasalahan pada penelitian ini:

a. Pembahasan tentang perancangan modul *login* dan modul registrasi atau modul pembuatan hak akses menggunakan PHP.
b. Pengujian modul terhadap keamanan informasi dari aplikasi *keylogger* dan *sniffer*.

1.4. Tujuan Penelitian

Tujuan yang ingin dicapai adalah merancang modul *login* dan modul registrasinya yang mampu mencegah pemanfaatan data akun dari aplikasi *keylogger* dan *sniffer*.

1.5. Sistematika Penulisan

Berikut merupakan rencana susunan sistematika penulisan laporan tugas akhir yang akan dibuat :

BAB I PENDAHULUAN

Bagian ini berisi tentang deskripsi umum tugas akhir yang meliputi latar belakang masalah, perumusan masalah, tujuan penelitian, batasan penelitian, serta sistematika penulisan.

BAB II LANDASAN TEORI

Bagian ini menjelaskan tentang teori-teori umum, teori-teori khusus yang berhubungan dengan tugas akhir ini.

BAB III METODOLOGI PENELITIAN

Bagian ini menjelaskan tentang metode pengembangan modul, pengumpulan data dan objek, tempat serta waktu penelitian.

BAB IV ANALISA DAN PERANCANGAN

Bagian ini berisi tentang analisis dan perancangan modul. Pada tahap perancangan akan dibahas tentang perancangan modul.

BAB V IMPLEMENTASI DAN PENGUJIAN

Pada bagian implementasi dan pengujian berisi pembahasan mengenai implementasi modul *login* disertai dengan *preview* tampilan *interface* serta pengujian modul.

BAB VI KESIMPULAN DAN SARAN

Bagian ini berisi kesimpulan hasil penelitian beserta saran-saran yang berkaitan dengan penelitian ini.

BAB II

LANDASAN TEORI

Penyusunan tugas akhir ini membahas tentang perancangan dan implementasi yang dapat mencegah pemanfaatan identitas digital oleh orang lain yang menggunakan aplikasi *keylogger* dan *sniffer*. Sehingga pembahasan teori yang mendukung isi dari tugas akhir ini mengenai *keylogger* dan *sniffer* khususnya mengenai langkah kerja masing-masing aplikasi tersebut.

2.1. Keylogger

Pembahasan mengenai pengertian *keylogger*, cara kerja dan cara pencegahannya dijelaskan pada sub bab berikut ini.

2.1.1. Pengertian *Keylogger*

Keystroke logging (often called keylogging) is the practice of noting (or logging) the keys struck on a keyboard, typically in a covert manner so that the person using the keyboard is unaware that their actions are being monitored (*keystroke logging* atau *keylogging* adalah pencatatan penekanan tombol pada *keyboard*, pada umumnya dilakukan secara tersembunyi sehingga orang tidak akan tahu bahwa aktifitasnya sedang dipantau) (www.wikipedia.org,2009).

2.1.2. Cara Kerja Keylogger

Dari pengertian *keylogger* di atas dapat disimpulkan secara umum *keylogger* berfungsi untuk merekam atau pencatat setiap tombol *keyboard* yang

ditekan. Selain *keyboard*, *keylogger* juga dapat merekam penekanan *mouse*, menyalin data pada *clipboard* dan beberapa fitur lainnya.

2.1.3. Mencegah *Keylogger*

Beberapa cara untuk mencegah *keylogger* antara lain dengan menggunakan *virtual keyboard*. Dengan *virtual keyboard*, pengguna komputer tidak menggunakan *keyboard* untuk menuliskan *password*, tetapi cukup menekan tombol *keyboard* yang ada pada layar monitor (www.wikipedia.org, 2009).

Selain itu, *keylogger* juga dapat dicegah dengan menuliskan *password* secara acak, namun cara ini cukup merepotkan (www.wikipedia.org, 2009).

2.2. Sniffer

Pembahasan mengenai pengertian *sniffer*, cara kerja dan cara pencegahannya dijelaskan pada sub bab berikut ini.

2.2.1. Pengertian Sniffer

Sniffers are programs that passively monitor and capture network traffic (*sniffer* merupakan program yang memantau dan mengambil data pada jaringan secara pasif) (Klevinsky, T. J. dkk, 2002). Secara pasif maksudnya ialah *sniffer* tersebut tidak memberikan respon ke jaringan bahwa *sniffer* telah menerima paket data.

Hampir semua laptop dan PC dapat digunakan sebagai *sniffer* dengan cara menginstalkan aplikasi *sniffer* pada komputer tersebut. Aplikasi *sniffer* sangat banyak beredar di internet dan dapat di *download* secara bebas. Ini memungkinkan semua orang dapat melakukan *sniffing*.

Sniffer memanfaatkan teknologi *broadcast* pada jaringan, yaitu dimana data untuk satu komputer dapat dibaca oleh komputer lain.

Beberapa contoh aplikasi *sniffer* adalah Dsniff, Linsniff, Tcpdump, BUTTSniffer, dan lain-lain.

2.2.2. Cara Kerja *Sniffer*

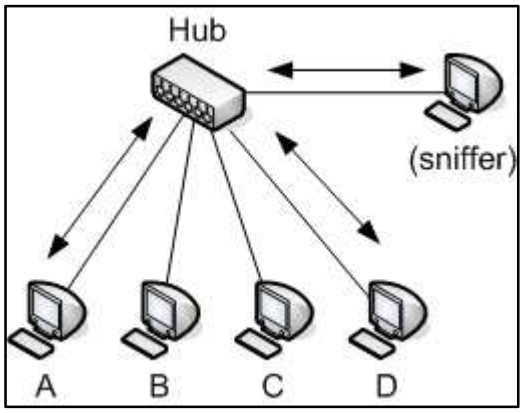
Komputer yang terhubung ke LAN (*Local Area Network*) memiliki dua alamat jaringan (*address*). Yang pertama MAC (*Media Access Control*) *Address* yang berbeda pada setiap titik (*node*) di dalam jaringan dan tersimpan di dalam kartu jaringan itu sendiri. Alamat MAC ini digunakan oleh protokol *ethernet* pada saat membuat *frame* untuk mengirim paket data dari atau ke mesin lain. Yang kedua adalah IP (*Internet Protocol*) *Address* yang biasa digunakan oleh aplikasi (Dhar, Summit, 2002).

Layer *Data Link* lebih menggunakan MAC *Address* dari pada IP *Address* pada *ethernet header*. Sedangkan layer *Network* bertanggung jawab untuk *mapping* IP *Address* ke MAC *Address*, seperti yang diperlukan oleh layer *Data Link* (Dhar, Summit, 2002).

Ada dua tipe dasar *Ethernet environment*, yaitu *Shared Ethernet* dan *Switched Ethernet* (Dhar, Summit, 2002).

a. Pada *Shared Ethernet Environment*, semua *host* terhubung ke jalur (*bus*) yang sama dan saling berbagi *bandwidth*. Ini berarti sebuah paket data yang ditujukan kepada salah satu *host* dapat diterima oleh *host* lainnya. Namun dalam kondisi normal, paket yang ditujukan untuk komputer lain akan di abaikan oleh komputer tersebut.

b. Sedangkan *Switched Ethernet* dapat mengelola tabel yang mengatur alamat MAC setiap komputer dan *physical port* pada *switch* dimana MAC *Address* tersebut dihubungkan. Sehingga sebuah paket akan dikirim hanya kepada komputer yang dituju dan tidak di-*broadcast* ke semua komputer pada jaringan tersebut.



Gambar 2.1. Cara Kerja *Sniffer*

2.2.3. Mencegah *Sniffer*

Sniffer dapat bekerja dengan baik pada jaringan yang terhubung dengan *hub*. Namun *sniffer* sulit digunakan pada *switch*. Apabila jaringan tersebut menggunakan *switch*, sniffer akan kesulitan untuk membaca data untuk komputer lain karena data yang sampai ke komputer tersebut telah di-*switch* (Dhar, Summit, 2002).

2.3. Look Up Table

In computer science, a lookup table is a data structure, usually an array or associative array, often used to replace a runtime computation with a simpler array indexing operation. The savings in terms of processing time can be significant, since retrieving a value from memory is often faster than undergoing an 'expensive' computation. Lookup tables are also used extensively to validate input values by matching against a list of valid (or invalid) items in an array and, in some programming languages, may include pointer functions (or offsets to labels) to process the matching input (Di bidang ilmu komputer, tabel *lookup* adalah sebuah struktur data, yang biasanya berupa *array* atau kumpulan *array*, sering digunakan untuk menggantikan perhitungan komputasi pada saat operasi berjalan dengan operasi penyusunan *array* yang sederhana. Penghematan dalam lingkup waktu pemrosesan bisa terjadi signifikan, karena menerima data dari memori lebih cepat daripada melakukan perhitungan yang ‘berat’. Secara luas, tabel *lookup* juga digunakan untuk memvalidasi nilai *input* dengan cara mencocokkan nilai terhadap daftar yang *valid* (atau tidak *valid*) dalam sebuah *array*, dan dalam beberapa bahasa pemrograman juga disertakan fungsi *pointer* (atau sejenis label) untuk memproses pencocokan *input*) (www.wikipedia.org, 2009).

Contoh sederhana manfaat lookup table untuk mengurangi waktu perhitungan komputasi adalah untuk mendapatkan hasil dari perhitungan trigonometri, seperti sin. Perhitungan trigonometri dapat memperlambat proses perhitungan pada sebuah aplikasi. Aplikasi yang sama dapat menyelesaikan

perhitungan tersebut dengan lebih cepat jika dilakukan pre-calculates (menghitung terlebih di awal) semua nilai sin. Misalkan telah dihitung terlebih dahulu nilai sin setiap 1 derajat dari 0 sampai 359 derajat. Apabila dibutuhkan nilai sin, maka tidak perlu dilakukan perhitungan sin, tetapi cukup mengambil dari nilai yang telah dihitung sebelumnya. Tentunya ini akan lebih mempercepat proses, karena mengambil sebuah nilai dari memory lebih cepat dibandingkan melakukan perhitungan untuk menghasilkan nilai tersebut (www.wikipedia.org, 2009).

Terdapat dua batasan yang fundamental pada saat mengkonstruksi sebuah *lookup table* untuk suatu proses yang diperlukan. Pertama adalah jumlah memori yang tersedia. Tidak bisa membuat suatu *lookup table* yang berukuran lebih besar dari ruang yang disediakan. Kedua adalah waktu yang dibutuhkan untuk menghitung nilai dari *lookup table* pada *instance* pertama, meskipun proses ini hanya perlu dan biasanya dilakukan satu kali, jika proses tersebut membutuhkan waktu yang cukup lama akan menyebabkan penggunaan *lookup table* menjadi tidak tepat. Bagaimanapun tabel dapat didefinisikan secara statik pada banyak kasus, ini dilakukan untuk menghindari pemrosesan tambahan pada saat telah di-*compile* (www.wikipedia.org, 2009).

Storage caches (termasuk disk caches pada file, atau processor caches untuk kode atau data) juga bekerja seperti lookup table. Tabel tersebut dibuat dengan memori yang cepat bahkan disimpan pada external memori yang lambat.

Dalam logika digital, sebuah n-bit dari lookup table dapat diimplementasikan dengan sebuah multiplexer yang memilih jalur sebagai input dari LUT (lookup table).

2.4. PHP

PHP dikenal sebagai sebuah bahasa *scripting* yang menyatu dengan *tag-tag* HTML, dieksekusi di *server* dan digunakan untuk membuat halaman *web* dinamis. Versi pertama PHP dibuat oleh Rasmus Lerdorf pada tahun 1995, berupa sekumpulan *script* PERL yang digunakan oleh Rasmus Lerdorf untuk membuat halaman *web* yang dinamis pada *homepage* pribadinya.

Berdasarkan hasil *survey* Netcraf pada bulan Desember 1999, lebih dari satu juta *site* yang menggunakan PHP, termasuk perusahaan besar seperti Mitsubishi, Redhat, NASA, Ericson dan banyak lagi. Berdasarkan *survey* Esoft pada bulan November 1999, 23% pengguna Apache Server menggunakan PHP (55% *webserver* dunia menggunakan Apache).

Dasar pertimbangan untuk mengembangkan kemampuan *object oriented* dari PHP adalah perkembangan aplikasi web sebagai sebuah *platform* yang terus meluas dengan cepat karena ruang lingkup aplikasi web semakin luas, sehingga aplikasi web yang dibangun juga menjadi semakin besar, rumit, dan kompleks.

Dengan kemapuan untuk mengimplementasikan konsep *object oriented*, PHP akan lebih mudah dapat berhubungan dan mengakses komponen-komponen yang dibuat dengan bahasa *object oriented* lain seperti Java atau C++ dan bahkan Visual Basic.

Berikut ini akan dijelaskan beberapa keunggulan PHP (Farid Azis, M, 2001):

a. *Life cycle* yang singkat, sehingga PHP selalu *up to date* mengikuti perkembangan teknologi internet.

b. *Cross platform*, PHP dapat dipakai di hampir semua *web server* yang ada di pasaran (Apache, fhttpd, phttpd, Microsoft IIS dan lain-lain) yang dijalankan pada berbagai sistem operasi. Dengan demikian proses *developing* dapat dilakukan menggunakan sistem operasi yang digunakan setelah *publish* (misalnya, pengembangan di Windows kemudian dipasang di *web server* yang menggunakan sistem operasi Linux).
c. PHP mendukung banyak paket *database* baik yang komersil maupun yang non komersil.

2.5. Analisis dan Perancangan Berorientasi Objek

Teknologi objek menganalogikan sistem aplikasi seperti kehidupan nyata yang didominasi oleh objek. Didalam membangun sistem berorientasi objek akan menjadi lebih baik apabila langkah awalnya didahului dengan proses analisis dan perancangan yang berorientasi objek. Tujuannya adalah untuk mempermudah *programmer* didalam mendesain program dalam bentuk objek-objek dan hubungan antar objek tersebut untuk kemudian dimodelkan dalam sistem nyata (A.Suhendar, 2002).

Suatu perusahaan *software*, Rational Software, telah membentuk konsorsium dengan berbagai organisasi untuk meresmikan pemakaian *Unified Modelling Language* (UML) sebagai bahasa standar dalam *Object Oriented Analysis Design* (O*OAD).*

2.5.1. *Unified Modelling Language* (UML)

UML adalah sebuah bahasa untuk menentukan, visualisasi, kontruksi, dan mendokumentasikan *artifacts* dari sistem *software*, untuk memodelkan bisnis, dan sistem *non-software* lainnya. UML merupakan sistem arsitektur yang bekerja dalam OOAD dengan satu bahasa yang konsisten untuk menentukan, visualisasi, mengkontruksi, dan mendokumentasikan *artifact* yang terdapat dalam sistem (A.Suhendar, 2002). *Artifact* adalah sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa *software*. *Artifact* dapat berupa model, deskripsi atau *software*.

Untuk membuat sebuah model, UML memiliki diagram grafis sebagai berikut:

2.5.2. *Use Case Diagram*.

Use case diagram menjelaskan manfaat sistem jika dilihat menurut pandangan orang yang berada diluar sistem (*actor*). Diagram ini menunjukkan fungsionalitas suatu sistem berinteraksi dengan dunia luar. *Uses case diagram* dapat digunakan selama proses analisis untuk menangkap *requirement* sistem dan untuk memahami bagaimana sistem seharusnya bekerja.

2.5.3. *Class Diagram*

Class diagram menjelaskan dalam visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. *Class diagram* mempelihatkan hubungan antar kelas dan penjelasan detail tiap-tiap kelas didalam model desain dari suatu sistem.

Selama proses analisis, class diagram memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.

2.5.4. Behavior Diagram

Behavior diagram dapat dikelompokkan menjadi 3 diagram, yaitu:

a. Statechart Diagram

Statechart diagram berfungsi untuk memodelkan perilaku dinamis satu kelas atau objek. Statechart diagram khususnya digunakan untuk memodelkan tarap-tarap diskrit dari sebuah siklus objek, sedangkan *activity diagram* paling cocok digunakan untuk memodelkan urutan aktivitas dalam suatu proses.

b. Activity Diagram

Activity diagram memodelkan alur kerja (*workflow*) sebuah proses bisnis dan urutan aktivitas dalam suatu proses. Diagram ini dapat memodelkan sebuah alur dari satu aktivitas ke aktivitas lainnya atau dari satu aktivitas kedalam keadaan sesaat (*state*). Membuat sebuah *activity diagram* dalam memodelkan sebuah proses akan dapat membantu dalam memahami proses secara keseluruhan.

c. Interaction Diagram

Interaction diagram di bagi menjadi 2 model diagram yaitu:

1. *Sequence diagram* menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus berasosisasi dengan *use cases*. *Sequence diagram* memperlihatkan tahap demi tahap

apa yang seharusnya terjadi untuk menghasilkan sesuatu didalam *use case*.

2. *Collaboration diagram* melihat pada interaksi dan hubungan terstruktur antar objek. Tipe diagram ini menekankan pada hubungan (*relationship*) antar objek, sedangkan *sequence diagram* menekankan pada urutan kejadian. Dalam *collaboration diagram* terdapat beberapa objek, *link*, dan *message*.

2.5.5. Implementation Diagram

Implementation diagram di bagi menjadi 2 diagram, yaitu:

1. *Component diagram* menggambarkan alokasi semua kelas dan objek kedalam komponen-komponen dalam desain fisik sistem *software*. Diagram ini memperlihatkan pengaturan dan kebergantungan antara komponen-komponen *software*, seperti *source code*, *binary code*, dan komponen tereksekusi (*execute components*).
2. *Deployment diagram* memperlihatkan pemetaan *software* kepada *hardware*. Dimana akan berjalan (di *server*/*multitier*, *standalone* atau lainnya), dan menggambarkan model koneksi dan kemampuan jaringan dan hal lainnya yang bersifat fisik.

2.6. *Rational Unified Process* (RUP)

Untuk pengembangan modul *Login* pada Tugas Akhir ini menggunakan *Rational Unified Process*.

2.6.1. Pengertian RUP

The Rational Unified Process® is a Software Engineering Process. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end-users, within a predictable schedule and budget (*Rational Unified Process* adalah sebuah Proses Rekayasa Perangkat Lunak. RUP menyediakan pendekatan disiplin untuk memberikan tugas dan tanggung jawab dalam organisasi pengembang perangkat lunak. Tujuannya untuk memastikan perangkat lunak yang berkualitas tinggi dan sesuai kebutuhan penggunanya dalam anggaran dan jadwal yang dapat diprediksi) (Ivar Jacobson, dkk, 2002).

RUP menjelaskan bagaimana cara mengembangkan sebuah perangkat lunak dengan efektif. Terdapat 6 *best practices* atau juga disebut sebagai *basic priciples,* antara lain :

a. *Develop software iteratively*, untuk mengurangi resiko di awal proyek.
b. *Manage requirements*.
c. *Employ a componen-based architecture*, untuk membangun arsitektur komponen dengan cepat.
d. *Model software visually*, merancang model visual sebuah perangkat lunak untuk mendapatkan struktur dan perilaku (*behavior*) dari arsitektur dan komponen.
e. *Continuosly verify quality*, pengecekkan terhadap *reliability*, *functionality*, *application performance* dan *system performance*.

f. *Control changes*, kemampuan untuk mengatur perubahan dan pemutakhiran terhadap perangkat lunak.

2.6.2. Fase RUP

Fase-fase pada RUP berdasarkan waktu pengerjaan proyek dapat dibagi menjadi 4 fase, yaitu *Inception, Elaboration, Construction* dan *Transition* (Rational Team, 2001).

a. Fase *Inception,* mengidentifikasi kasus atau permasalahan. Ini termasuk mengidentifikasi entitas yang berasal dari luar yang akan berinteraksi dengan sistem. Selain itu juga termasuk kriteria keberhasilan proyek, perkiraan resiko, perkiraan terhadap *resource* yang dibutuhkan dan merencanakan penjadwalan *milestone*. Hasil yang diperoleh pada fase ini adalah :

   i. Dokumen visi (visi dari kebutuhan projek, kata kunci, batasan utama).
   ii. Inisialisasi model *use-case* (10%-20% selesai).
   iii. *Project Glossary* (daftar kata).
   iv. Inisialisasi *business-case* (proyeksi pendapatan, melihat kondisi pasar) dan *financial forecast* (ramalan keuangan).
   v. Inisialisasi *risk assessment* (penilaian resiko).
   vi. Rencana proyek (*project plan*)
   vii. *Business model*.
   viii. Beberapa contoh prototipe

Kriteria evaluasi untuk fase *Inception* adalah :

i. Menyesuaikan *stakeholder* dengan *scope definition* dan perkiraan biaya atau perkiraan jadwal.
ii. Pemahaman terhadap *use-case* utama
iii. Kredibilitas dari perkiraan biaya, jadwal, prioritas, resiko dan proses pengembangan.
iv. Pemahaman terhadap *prototype*

b. Fase *Elaboration*, tujuan dari fase *Elaboration* (pengembangan) adalah untuk menganalisa *domain* (area) permasalahan, mengembangkan rencana proyek dan mengurangi resiko-resiko terbesar dalam proyek. Fase ini memastikan bahwa *requirement* dan rencana proyek cukup stabil serta meminimalkan tingkat resiko. Hasil yang didapat dari proses *Elaboration* ini adalah :

i. Model *use-case* (minimal 80% selesai) : semua *use-case* dan *actor* telah teridentifikasi.
ii. *Requirement* tambahan yang mungkin tidak bersifat fungsional bagi proyek.
iii. *Software Architecture Description* (Deskripsi Arsitektur Perangkat Lunak).
iv. Prototipe dari arsitekture yang dapat dieksekusi.
v. Revisi daftar tingkat resiko dan revisi *business-case*.
vi. Rencana pengembangan keseluruhan proyek.

vii. Persiapan dokumen panduan bagi pengguna (*user manual*).

Kriteria evaluasi terhadap fase *Elaboration* ini adalah :

i. Apakah dari produk telah kuat dan stabil?

ii. Apakah arsitektur proyek telah kuat dan stabil?

iii. Apakah demonstrasi prototipe telah menunjukkan bahwa tingkat resiko telah atur dan dapat diatasi?

iv. Apakah rencana konstruksi telah detail dan akurat? Apakah itu telah di-*backup* dengan dasar yang dapat dipercaya?

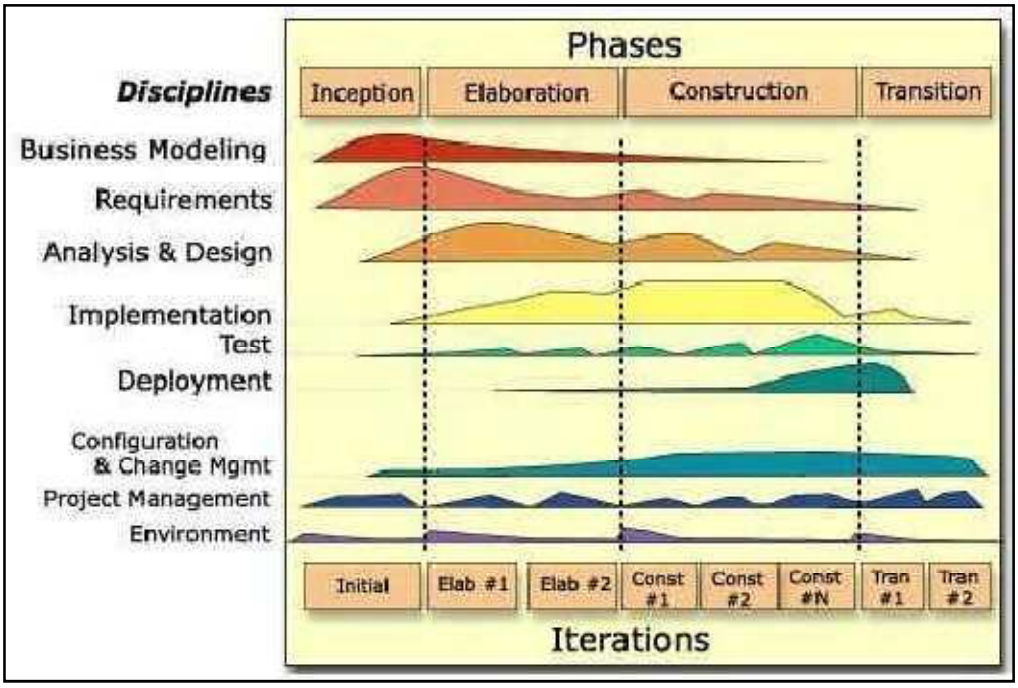
v. Apakah *stakeholder* bersedia dan menyepakati visi dari pengembangan proyek tersebut?

vi. Apakah pembelanjaan *actual-resource* terhadap rencana pembelanjaan dapat diterima?

c. Fase *Construction*, selama tahap ini semua komponen dan fitur dari aplikasi dikembangkan dan terintegrasi ke dalam produk dan telah melewati proses pengujian. Di lain sisi, proses konstruksi adalah sebuah proses *manufacturing*, dimana terdapat penekanan dalam mengelola *resource* dan mengatur operasi untuk mengoptimalkan jadwal dan kualitas. Pada tahap ini pola pikir (*mindset*) mengalami perubahan dari pengembangan *intellectual property* pada fase *Inception* dan *Elaboration*, menjadi pengembangan *deplyoable product*. Kriteria evaluasi terhadap fase *Construction* ini adalah :

i. Apakah peluncuran produk cukup baik dan dapat diterima di komunitas pengguna?
ii. Apakah semua *stakeholder* siap untuk beralih ke komunitas pengguna?
iii. Apakah pembelanjaan *actual-resource* terhadap rencana pembelanjaan masih tetap diterima?

d. Fase *Transition*, tujuan dari fase ini adalah untuk proses peralihan dari produk perangkat lunak ke komunitas pengguna (*end-user*). Apabila produk telah diberikan kepada komunitas pengguna, maka isu-isu akan muncul dari berbagai pengguna. Isu ini yang akan digunakan untuk mengembangkan perangkat lunak versi baru, mengoreksi kesalahan atau fitur-fitur yang belum sempurna. Kriteria evaluasi untuk fase *Transition* adalah :
   i. Apakah pengguna merasa puas?
   ii. Apakah pembelanjaan *actual-resource* terhadap rencana pembelanjaan masih tetap diterima?



Gambar 2.2. RUP *lifecycle*[1]

2.7. *Message Digest* (MD5)

Penggunaan enkripsi pada modul *login* akan menggunakan algoritma *Message Digest* (MD5). Berikut akan dijelaskan tentang MD5.

2.7.1. Sekilas Tentang MD5

Dalam kriptografi, MD5 (*Message-Digest algortihm 5*) ialah fungsi *hash* kriptografik yang digunakan secara luas dengan *hash value* 128-bit. Pada standar Internet (RFC 1321), MD5 telah dimanfaatkan secara bermacam-macam pada aplikasi keamanan, dan MD5 juga umum digunakan untuk melakukan pengujian integritas sebuah file (wikipedia.org, 2009).

MD5 di desain oleh Ronald Rivest pada tahun 1991 untuk menggantikan *hash function* sebelumnya, MD4. Pada tahun 1996, sebuah kecacatan ditemukan

[1] *Copyright* 1999-2005 IBM

dalam desainnya, walau bukan kelemahan fatal, pengguna kriptografi mulai menganjurkan menggunakan algoritma lain, seperti SHA-1 (klaim terbaru menyatakan bahwa SHA-1 juga cacat). Pada tahun 2004, kecacatan-kecacatan yang lebih serius ditemukan menyebabkan penggunaan algoritma tersebut dalam tujuan untuk keamanan jadi makin dipertanyakan (wikipedia.org, 2009).

2.7.2. Algoritma MD5

Berdasarkan Rivest (RFC1321) algoritma MD5 dibagi menjadi 5 tahap, yaitu :

a. *Append Padding Bits* (Menambahkan bit tambahan)

   Pada tahap ini pesan akan ditambahkan sebuah bit bernilai “1” pada akhir dan menambahkan beberapa bit “0” sehingga ukuran pesan menjadi 448 (modulo 512).

b. *Append Length* (Menambahkan informasi ukuran pesan)

   Pada tahap ini pesan akan ditambahkan informasi tentang ukuran pesan sebelum ditambahkan bit tambahan. Informasi tersebut berupa angka biner dengan ukuran 64 bit. Sehingga panjang pesan genap menjadi 512 (modulo 512).

c. *Initialize MD Buffer* (Inisialisasi MD Buffer)

   Pada tahap ini akan ditentukan MD Buffer berupa 4 buah kata yang digunakan untuk menghitung MD5. Masing-masing kata terdiri dari 32 bit. Jika dikonversi ke heksadesimal, maka MD Buffer tersebut adalah:

   ```
   Kata A = 01 23 45 67
   Kata B = 89 ab cd ef
   ```

Kata C = fe dc ba 98

Kata D = 76 54 32 10

d. *Process Message in 16-Word Block* (Proses Pesan dalam Blok 16-Kata)

Tahap ini merupakan bagian utama dalam proses MD5. Sebelum dilakukan komputasi terhadap pesan, maka harus ditentukan fungsi pembantu berikut ini :

F(X,Y,Z) = XY v (-X)Z

G(X,Y,Z) = XZ v Y(-Z)

H(X,Y,Z) = X xor Y xor Z

I(X,Y,Z) = Y xor (X v (-Z))

Selanjutnya dibangkitkan sebuah tabel dengan 64 elemen yang dibangkitkan dari fungsi sin.

For i from 0 to 63

K[i] := floor(abs(sin(I + 1)) x 2^{32})

Setelah itu dilakukan proses komputasi terhadap pesan (Lampiran A).

e. *Output* (Hasil)

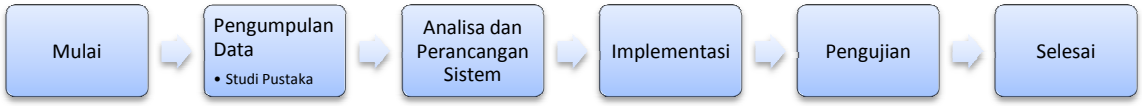
Setelah dilakukan komputasi maka akan dihasilkan A, B, C, D. (Lampiran A)

BAB III

METODOLOGI PENELITIAN

3.1. Tahapan Penelitian

Tahapan penelitian yang akan dilaksanakan pada perancangan modul login ini dapat dilihat pada Gambar 3.1.



Gambar 3.1. Tahapan Penelitian

3.1.1. Pengumpulan Materi

Pengumpulan materi merupakan tahapan persiapan yang harus dilaksanakan terlebih dahulu sebelum dilakukan penelitian. Berikut merupakan aktivitas yang dilaksanakan dalam pengumpulan sumber teori :

a. Studi Literatur

   Berfungsi sebagai pendukung dari penelitian yang akan dilaksanakan. Teori-teori yang digunakan bersumber dari buku, jurnal dan penelitian-penelitian sejenis yang dapat mendukung pemecahan masalah dalam penelitian yang dilakukan.

Literatur yang dikumpulkan antara lain adalah :

a. *Keylogger* dan *sniffer*, mencakup definisi, cara kerja dan cara antisipasi terhadap aplikasi *keylogger* dan *sniffer*.

b. Bahasa pemrograman web PHP, mencakup sejarah, dan keunggulannya.
c. *Object Oriented Analysis and Design* (OOAD), mencakup deskripsi dan struktur OOAD.
d. *Rational Unified Process* (RUP), mencakup deskripsi dan fase-fase dari RUP.
e. *Message Digest* (MD5), mencakup sejarah dan algoritma MD5.

3.1.2. Analisa dan Perancangan Sistem

Tahapan ini mengenali seluruh permasalahan yang muncul dalam mengenali komponen-komponen sistem, objek-objek, hubungan antar objek, mempelajari prosedur sistem yang akan dibuat, serta menganalisis solusi maksimal atas kebutuhan sistem. Proses ini akan menggunakan pendekatan OOAD dalam menentukan model dan struktur modul.

3.1.3. Implementasi

Implementasi akan dilakukan pada tahap ini, yaitu setelah fase perancangan selesai. Pengkodean modul ini akan menggunakan pemrograman PHP versi 5. Karena PHP versi 5 telah mampu menerapkan *Object Oriented Programming* (OOP). PHP juga telah mendukung algoritma MD5, sehingga tidak diperlukan untuk membuat kode algoritma MD5. Ini dilakukan untuk membuktikan bahwa modul ini telah mampu berkerja dengan baik walaupun dengan algoritma *hash* yang sering digunakan atau bahkan tanpa enkripsi. Namun,

untuk mencegah pencurian informasi langsung dari *database*, maka diperlukan sebuah enkripsi untuk melindungi data di *database*.

3.1.4. Pengujian Sistem

Ini bertujuan untuk mengetahui apakah sistem yang dibangun sudah sesuai dengan kebutuhan. Apabila didalam *testing* tersebut terdapat kesalahan (*error*), maka harus dilakukan perbaikan kembali pada sistem tersebut. Sistem dapat terapkan pada lingkungan operasionalnya setelah semua fase pengujian sukses dilalui dan selanjutnya dapat dilakukan pengujian terhadap kemampuan sistem.

3.2. Waktu Penelitian

Tabel 3.1. Rencana Penyelesaian Tugas Akhir

No	Tahapan	Juni	Juli	Agt	Sept	Okt	Nov
1	Pengajuan Judul	√					
2	Studi Pustaka	√	√	√			
3	Analisa dan Perancangan Modul		√	√	√		
4	Implementasi dan Pengujian Modul			√	√	√	
5	Laporan	√	√	√	√	√	√
6	Revisi					√	√
7	Seminar Hasil dan Sidang						√

3.3. Tahapan RUP

Berikut ini akan diuraikan tahapan-tahapan pembuatan modul login menggunakan *Rational Unified Process* (RUP).

3.3.1. Fase *Inception*

Pada fase ini akan dilakukan tugas-tugas sebagai berikut:

a. Pengenalan masalah, yaitu memahami permasalahan yang terjadi.
b. Pembuatan proposal, yaitu mencakup latar belakang permasalahan, pokok permasalahan, tujuan dan batasan permasalahan.
c. Studi literatur, mencakup penelusuran teori-teori yang berhubungan dengan permasalahan.
d. *Project Plan*, yaitu mencakup jadwal pelaksanaan tugas-tugas yang akan dijalani.
e. Penyelesaian model *use-case* (10%-20%).
f. Contoh prototipe.

3.3.2. Fase *Elaboration*

Pada fase *elaboration* akan dilakukan tugas-tugas sebagai berikut:

a. Penyempurnaan model *use-case* (80%-100%).
b. Penyempurnaan *Project Plan*.
c. Pembuatan deskripsi arsitektur modul.

3.3.3. Fase *Construction*

Pada fase ini akan dilakukan tugas-tugas sebagai berikut:

a. Pengkodean modul yang berpedoman pada model *use-case* menggunakan bahasa pemrograman PHP.
b. Melakukan pengujian terhadap kesalahan-kesalahan yang mungkin akan terjadi selama proses pengkodean.

3.3.4. Fase *Transition*

a. Melakukan pengujian modul terhadap aplikasi-aplikasi *sniffer* dan *keylogger*.
b. Mengamati kekurangan yang ada.
c. Menyimpulkan fitur-fitur tambahan guna pengembangan modul untuk versi selanjutnya.

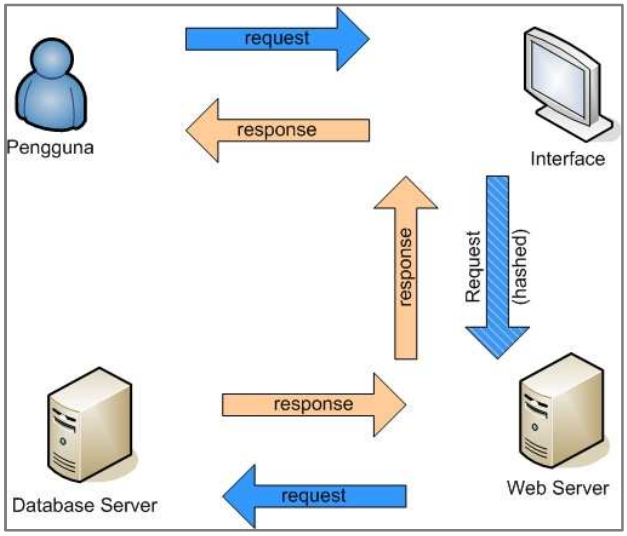
BAB IV

ANALISA DAN PERANCANGAN

Pada bab ini pembahasan akan dibagi menjadi tiga bagian, yaitu deskripsi umum, analisa modul dan perancangan modul.

4.1. Deskripsi Umum

Gambaran umum modul otentikasi yang dibuat adalah modul otentikasi yang digunakan pada situs *web* yang memerlukan keamanan terhadap pemanfaatan *username* dan *password* oleh orang lain. Contoh situs *web* yang memerlukan keamanan tersebut berupa situs *web* bank atau situs *web* lainnya. Modul otentikasi yang akan dibuat untuk selanjutnya akan disebut dengan *modAuth* (Modul *Authentication*). Ilustrasi terhadap model kerja *modAuth* secara umum dapat dilihat pada gambar 4.1.



Gambar 4.1 Model Kerja *modAuth* Secara Umum

Pada gambar 4.1 dapat dilihat bahwa Pengguna merupakan orang yang berinteraksi dengan *Interface*. *Interface* merupakan situs *web* yang menggunakan *modAuth* sebagai modul otentikasinya. Aktivitas dapat dilakukan oleh pengguna di antaranya:

1. Melakukan pendaftaran Pengguna baru (Mendaftarkan Pengguna Baru).
2. Melakukan proses *Log In*.
3. Mengubah data Pengguna, terdiri dari mengubah data pribadi, *password* dan *lookup table* – untuk selanjutnya akan disebut dengan nama *mCode*.
4. Melakukan proses *Log Out*.
5. Melakukan proses mendapatkan-kembali data pengguna, meliputi pengaturan-ulang terhadap *password* – untuk selanjutnya akan disebut *Reset Password* – dan perbaharuan *mCode* (*Reset mCode*).
6. Melakukan *hashing*(pengacakan) terhadap *password* dan *mCode* sebelum ditransfer melalui jaringan komputer menggunakan MD5 Algorithm dengan bahasa pemrograman JavaScript.

4.2. Analisa Modul Lama

Pembahasan analisa pada modul yang lama meliputi model kerja modul yang lama dan beberapa kelemahan pada modul yang lama.

4.2.1. Model Kerja Modul Lama

Secara umum, model kerja modul otentikasi yang digunakan, yaitu :

1. Pengguna memasukkan *username* dan *password* lalu menekan tombol *Login*. Jika pengguna tidak menggunakan aplikasi *virtual keyboard*, kondisi ini dapat dimanfaatkan untuk mencatat apa yang sedang diketik oleh pengguna menggunakan aplikasi *keylogger*. Cara mudah untuk mengacak hasil pencatatan aplikasi *keylogger* yaitu dengan mengacak rangkaian huruf atau angka yang akan ditulis, sehingga hasil pencatatan oleh *keylogger* juga akan teracak. Namun ini tentu akan mempersulit pengguna untuk mengacak setiap kali akan melakukan proses otentikasi. Selain itu, pengacakan tersebut juga memungkinkan dapat ditebak dengan menggunakan metode *brute-force*. Jika seseorang akan memasukkan *password*, contoh `affandes`. Pengguna dapat mengetik *password* tersebut secara acak dan dibantu dengan menggunakan *mouse*, seperti `andes`, lalu arahkan *pointer* ke awal kata dengan *mouse* dan ketik `aff`. Sehingga hasil pencatatan pada *keylogger* menjadi `andesaff`, bukan `affandes`.
2. Data *username* dan *password* akan melewati jaringan. Jika menggunakan protokol *HTTPS*, maka data akan melewati jaringan akan di-*encrypt*. Namun jika menggunakan protokol *HTTP*, maka data akan melewati jaringan tanpa di-*encrypt*. Kondisi ini dapat dimanfaatkan untuk mendapatkan data yang melewati jaringan tersebut dengan menggunakan aplikasi *sniffer*.

3. Selanjutnya data *username* dan *password* akan dibandingkan ke *database* sehingga didapat data pengguna tersebut.
4. Selain itu terdapat modul kerja otentikasi menggunakan *token* – sebuah alat khusus yang menghasilkan kode yang dapat digunakan untuk melakukan otentikasi – yang diterapkan di situs *web* bank.

4.2.2. Kelemahan Modul Lama

Beberapa kelemahan pada modul lama dapat diilustrasikan sebagai berikut:

1. Pada saat pengguna memasukkan *username* dan *password* secara acak pada modul lama, data *username* dan *password* tersebut juga akan tercatat sebagai kode acak pada aplikasi *keylogger*, namun tetap dapat dibaca oleh aplikasi *sniffer* jika situs *web* tersebut tidak menggunakan protokol *HTTPS*.
2. Atau, sebaliknya. Jika situs *web* tersebut menggunakan protokol *HTTPS*, tetapi pengguna tidak mengacak *username* dan *password* pada saat di-*input*-kan, ini dikarenakan kesulitannya pada saat mengacak *password* yang ditampilkan berupa tanda asteriks (*) pada saat diketik.
3. Kondisi terbaik adalah pengguna menggunakan *virtual keyboard* pada saat memasukkan *username* dan *password* ke situs *web* yang telah menggunakan protokol *HTTPS*, atau.
4. Pengguna menggunakan *token* untuk melakukan otentikasi.

Dari ilustrasi di atas bahwa kondisi (3) masih memiliki kelemahan. Ini dikarenakan *virtual keyboard* tidak menjadi bagian dari modul otentikasi yang

lama. *Virtual keyboard* merupakan aplikasi tambahan yang terpisah, seperti OnScreen Keyboard pada Windows XP atau aplikasi *keylogger* lainnya. Selain itu *virtual keyboard* juga akan menampilkan tuts *keyboard* di layar monitor. Kemudian untuk penggunaan *token*, pengguna akan dibebankan dengan pembayaran untuk membeli alat tersebut.

4.3. Analisa Modul Baru

Analis modul yang akan dibuat akan melihat dari beberapa kelemahan dari modul yang lama. Untuk mempermudah pemahaman dalam menganalisa modul baru, maka pembahasan akan dibagi menjadi beberapa bagian.

4.3.1. Analisa Kebutuhan

Kebutuhan yang diperhatikan dalam menganalisa modul baru adalah kebutuhan terhadap modul yang dapat menjaga keamanan data pengguna tanpa perlu khawatir terhadap pemanfaatan menggunakan aplikasi *keylogger* dan aplikasi *sniffer*.

Selain itu, ada beberapa kebutuhan lain terhadap modul otentikasi yang baru, yaitu:

1. Modul otentikasi yang langsung mengacak atau memaksa pengguna untuk mengacak kode yang diinputkan tanpa perlu melakukan pengacakan sendiri oleh pengguna. Sehingga kode acak tersebut tetap diinputkan oleh pengguna secara berurut.

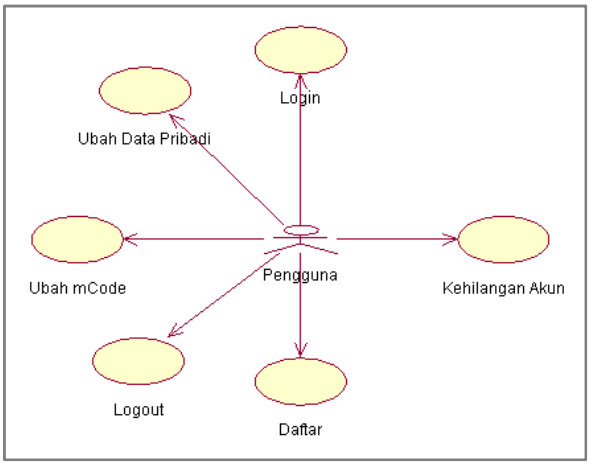
2. Modul otentikasi yang mengirimkan kode acak sehingga dapat mempersulit untuk mendapatkan kode sebenarnya jika berhasil didapatkan menggunakan aplikasi *sniffer*.

4.3.2. Analisa Fungsional Modul

Tahap analisa ini akan memaparkan model dari *modAuth* menggunakan UML, yang terdiri dari *use case diagram*, *activity diagram*, *sequence diagram*, *statechart diagram*.

1. Use Case Diagram

Use case diagram terdiri dari aktor yang merupakan pelaku pada *modAuth*. Aktor pada *modAuth* ini diberi nama Pengguna, yaitu pengguna dari *modAuth*. Untuk lebih jelasnya dapat dilihat pada gambar 4.2.



Gambar 4.2 *Use Case Diagram modAuth*

Keterangan mengenai *use case diagram modAuth* di atas dijelaskan pada tabel 4.1.

Tabel 4.1 Keterangan *Use Case Diagram modAuth*.

No	Nama Aktor / *Use Case*	Keterangan
1	Pengguna (*Actor*)	Pengguna yang akan melakukan proses otentikasi atau pendaftaran data pengguna baru.
2	Daftar (*Use Case*)	Proses untuk mendaftarkan atau menambah data pengguna yang baru.
3	*Login* (*Use Case*)	Proses untuk melakukan otentikasi.
4	Kehilangan Akun (*Use Case*)	Merupakan proses untuk memperoleh informasi akun akibat kehilangan *password* ataupun *mCode*.
5	Ubah Data Pribadi (*Use Case*)	Proses untuk mengubah data pengguna
6	Ubah *mCode (Use Case)*	Proses untuk membuat *mCode* baru yang dilakukan sedang *Login*.
7	*Logout (Use Case)*	Proses untuk mengakhiri sesi otentikasi.

2. Activity Diagram

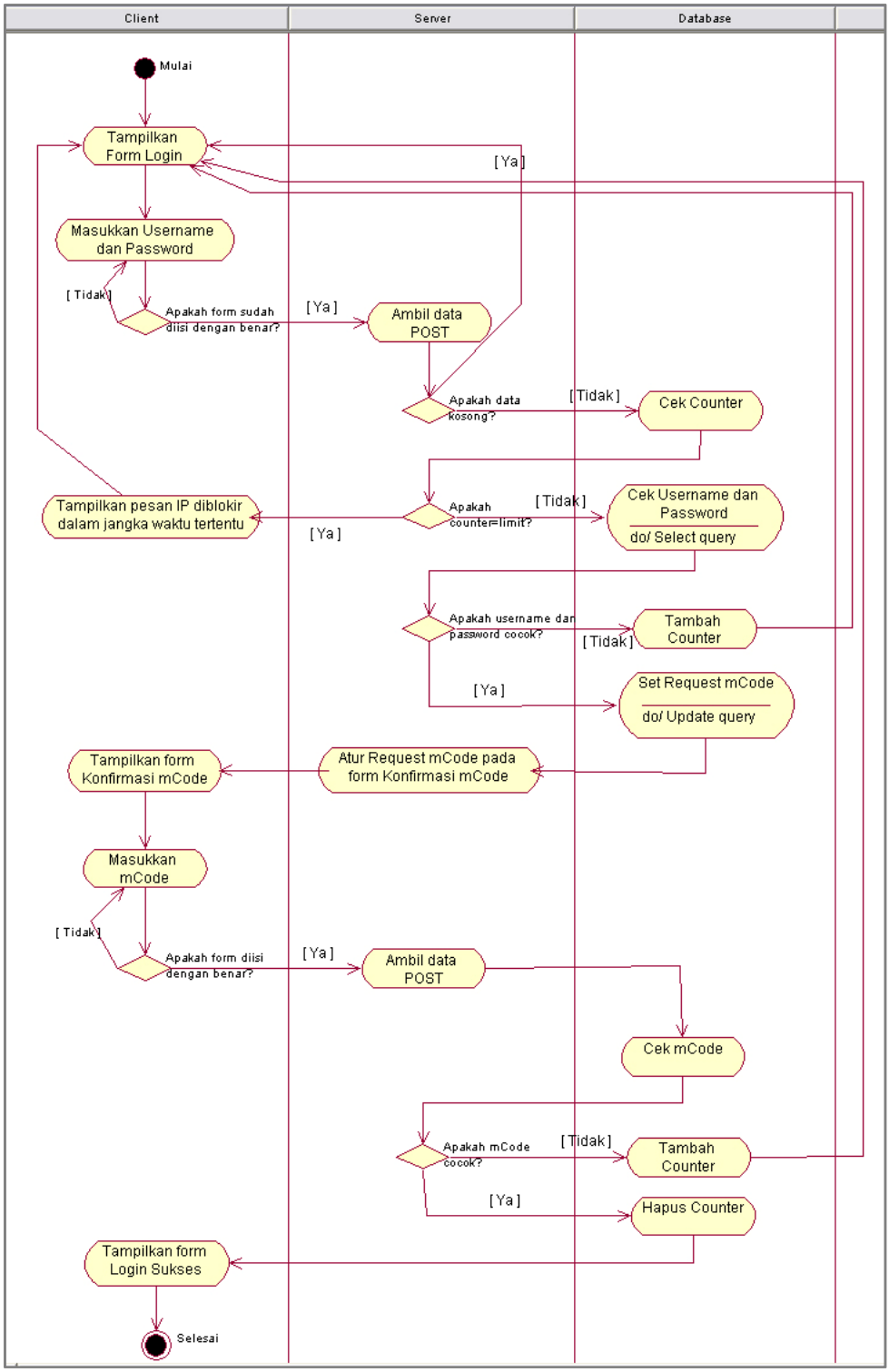
Aktivitas yang terjadi pada *modAuth* dapat dimodelkan dengan *activity diagram*. *Activity diagram* untuk proses *Login* dapat dilihat pada gambar 4.3.

Pada *activity diagram* proses *login* (gambar 4.3) dapat diketahui bahwa proses tersebut merupakan satu rangkaian untuk proses otentikasi. Untuk melakukan otentikasi, pengguna harus memasukkan *username* dan *password*.

Jika *username* dan *password valid*, maka akan di set *Request mCode*. *Request mCode* merupakan kode *mCode* yang akan diminta oleh *modAuth* pada saat Konfirmasi *mCode*.

Selanjutnya pengguna akan diminta memasukkan *mCode* berdasarkan *request mCode* dari *modAuth*. Jika *mCode* yang dimasukkan pengguna cocok dengan *mCode* yang ada di *database*, maka *login* sukses, jika tidak cocok, maka akan diulang ke tahap awal.

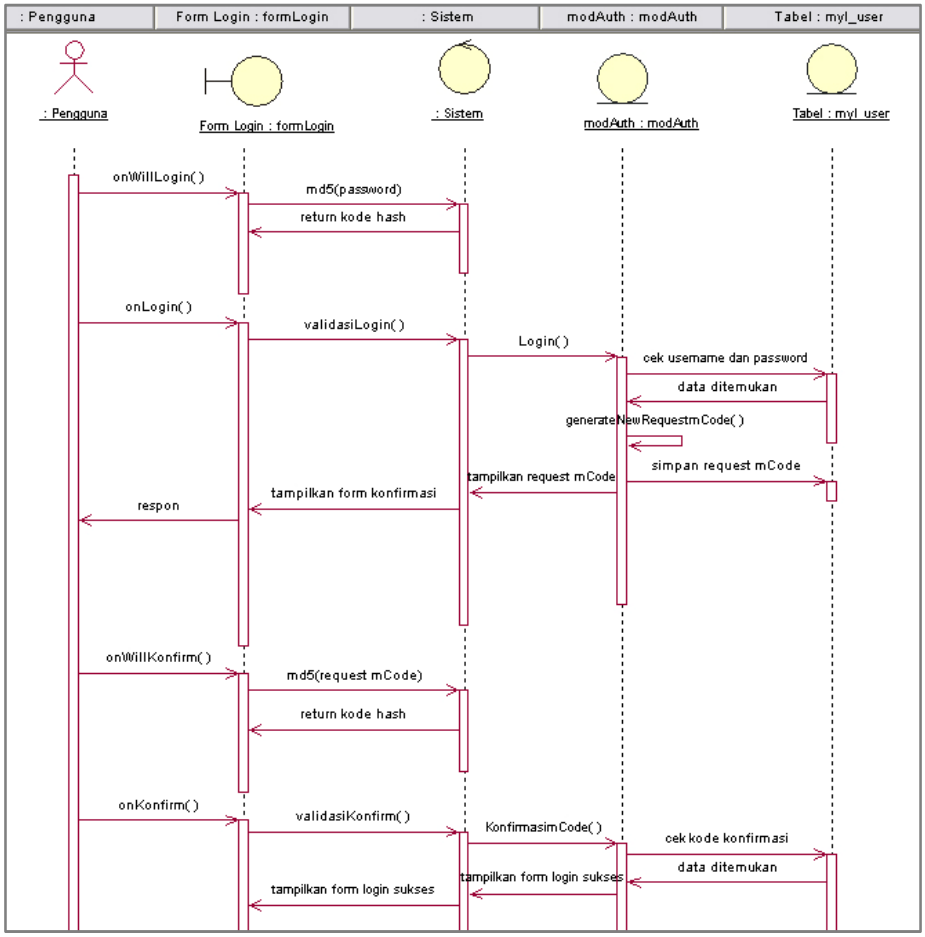
Activity diagram untuk proses lainnya dapat dilihat pada lampiran B.



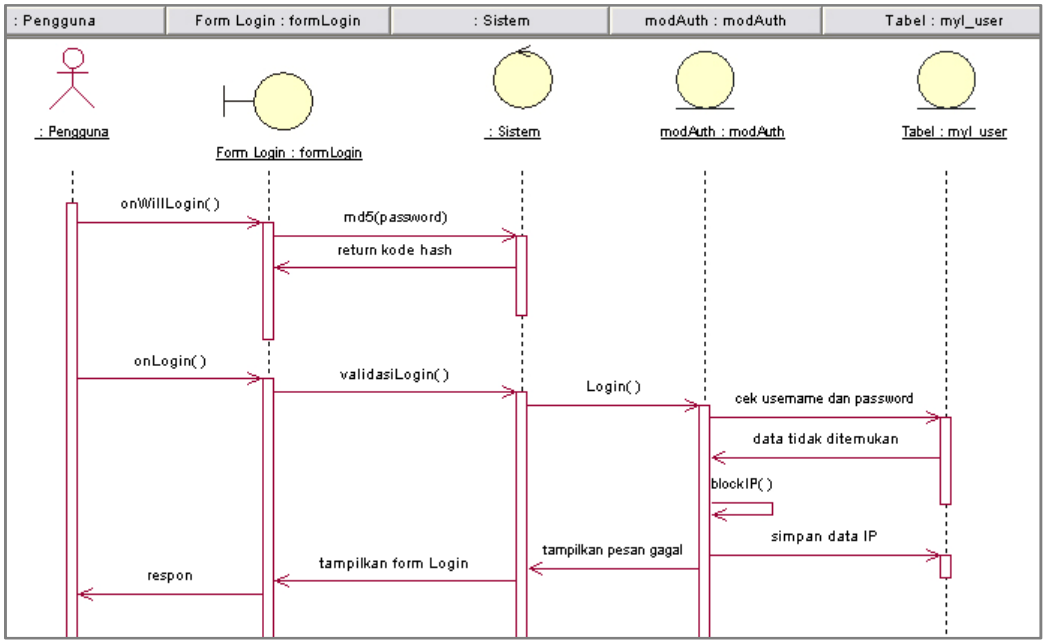
Gambar 4.3 ***Activity Diagram*** **Proses** ***Login*** **pada** ***modAuth***

3. Sequence Diagram

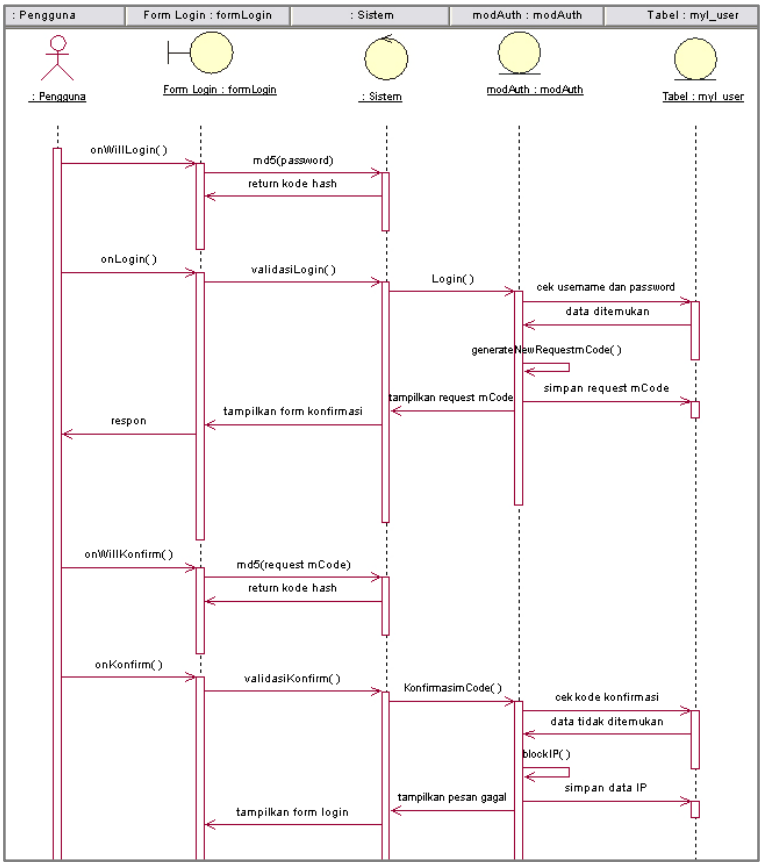
Urutan interaksi pada masing-masing *use case* berdasarkan waktu dapat dijelaskan pada *sequence diagram*. *Sequence diagram* ini terdiri dari diagram Aliran Normal (*normal flow*), diagram aliran pengecualian (*exception flow*) dan diagram aliran alternatif (*alternative flow*). Pada *sequence diagram* proses *Login*, terdapat satu diagram *normal flow* (gambar 4.4) dan dua diagram *exception flow* (gambar 4.5 dan gambar 4.6). *Sequence diagram* untuk proses lainnya dapat dilihat pada lampiran B.



Gambar 4.4 *Sequence Diagram* Proses *Login Normal Flow*



Gambar 4.5 *Sequence Diagram* Proses *Login Exception Flow (1)*



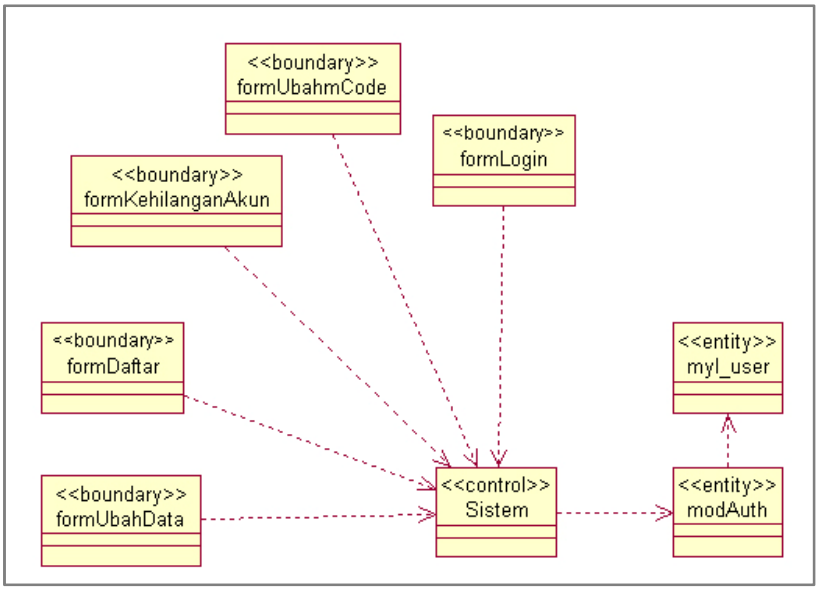
Gambar 4.6 *Sequence Diagram* Proses *Login Exception Flow (2)*

4.4. Perancangan Modul

Dalam tahap perancangan akan dibagi menjadi beberapa bagian pembahasan yaitu perancangan *class diagram*, *component diagram*, *database*, struktur modul dan perancangan *prototype user interface*.

4.4.1. Class Diagram

Berdasarkan diagram-diagran sebelumnya dapat menghasilkan sebuah *class diagram modAuth* (gambar 4.7). Pada *class diagram* tersebut tidak ditampilkan semua *attribute* dan *operation* pada *class*. Untuk melihat lebih detail tentang *class diagram modAuth* beserta *attribute* dan *operation*, dapat dilihat pada Lampiran B.



Gambar 4.7 *Class Diagram modAuth*

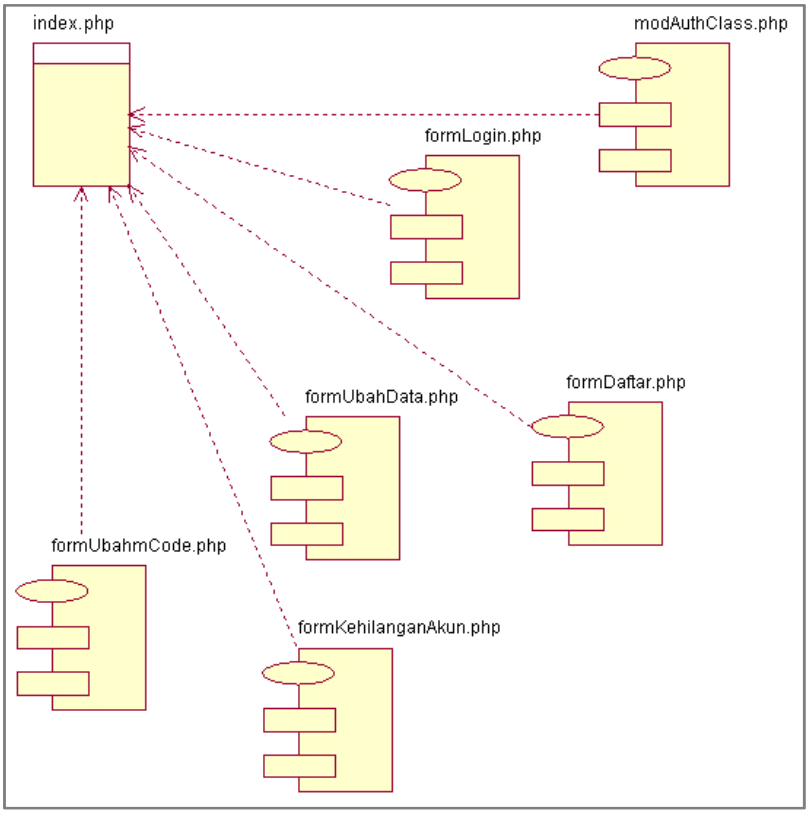
Berikut keterangan tentang *class diagram modAuth* (tabel 4.2).

Tabel 4.2 Keterangan *Class Diagram modAuth*

No	Nama *Class*	Keterangan
1	modAuth	*Class* utama yang merupakan *class* yang mengatur semua proses yang berhubungan dengan *class MySQL Database*
2	formDaftar	*Form* untuk menambah data pengguna baru
3	formLogin	*Form* untuk melakukan otentikasi
4	formKehilanganAkun	*Form* untuk mendapatkan kembali data pengguna yang hilang.
5	formUbahmCode	*Form* untuk membuat *mCode* yang baru
6	formUbahData	*Form* untuk mengubah data pengguna
7	myl_User	Merupakan *class* dari fungsi *MySQL Database.*
8	Sistem	Merupakan *class* yang mengatur *form* yang akan ditampilkan

4.4.2. Component Diagram

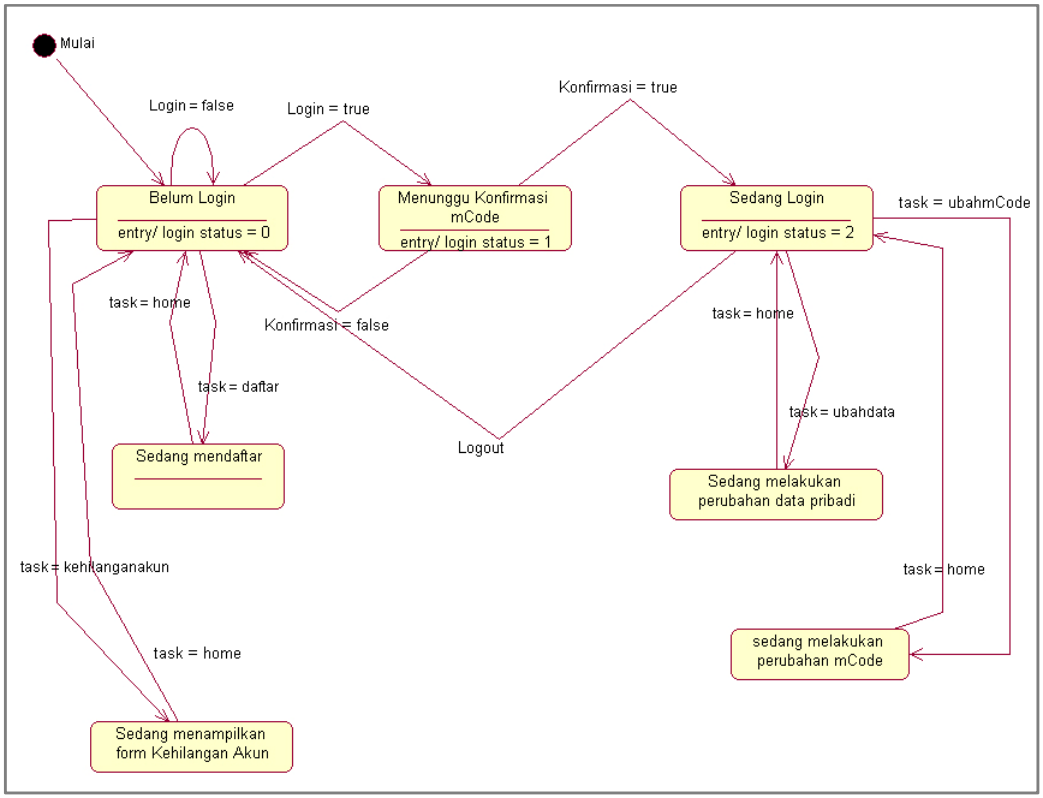
Berikut merupakan diagram yang menggambarkan struktur komponen dari *modAuth*.



Gambar 4.8 *Component Diagram modAuth*

4.4.3. Statechart Diagram

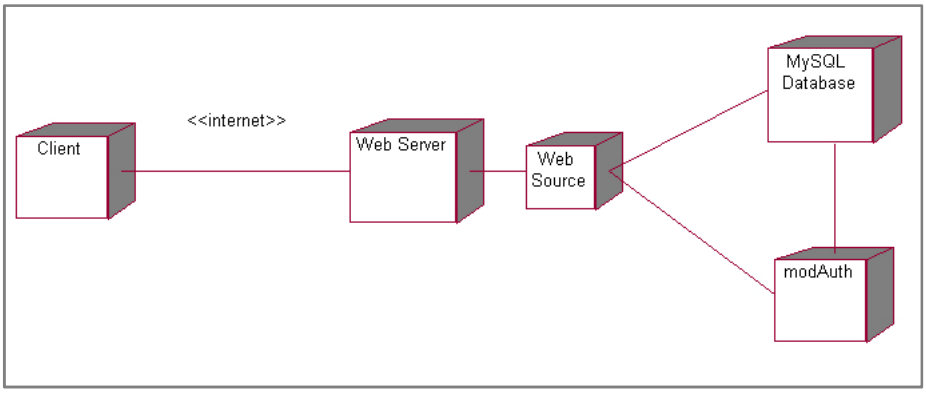
Perilaku dinamis dari masing-masing objek dapat dijelaskan dengan *statechart diagram*. *Statechart diagram* untuk *modAuth* secara umum dapat dilihat pada gambar 4.9.



Gambar 4.9 *Statechart Diagram modAuth*

4.4.4. Deployment Diagram

Deployment diagram menggambarkan struktur *package* dari *modAuth*. Pada diagram ini dapat dilihat posisi *modAuth* pada sebuah *website* (gambar 4.10).



Gambar 4.10 *Deployment Diagram modAuth*

4.4.5. Rancangan *Database*

Rancangan *database* untuk *modAuth* tidak memiliki relasi, ini dikarenakan bahwa *modAuth* sebenarnya hanya modul yang membutuhkan satu tabel. Struktur *database modAuth* dapat dilihat pada tabel 4.3.

Tabel 4.3 Keterangan Tabel *user* Pada *Database modAuth*

No	**Nama *Field***	**Tipe Data**	***Null***	**Keterangan**
1	Id (*PK*)	Int(11)	Tidak	Nomor registrasi pengguna.
2	Username (*Unique*)	Varchar(30)	Tidak	Nama yang digunakan untuk *Login*.
3	Password	Varchar(32)	Tidak	Kata sandi untuk otentikasi (*hashed*)
4	Passwordmustchange	Boolean	Tidak	Menandakan bahwa password dalam keadaan direset.
5	Datecreated	Datetime	Tidak	Tanggal registrasi pengguna.
6	Datemodified	Datetime	Tidak	Tanggal terakhir data diubah.
7	Firstip	Varchar(15)	Tidak	Alamat IP saat registrasi pengguna.
8	Lastlogin	Datetime	Tidak	Tanggal terakhir login.
9	Lastlogout	Datetime	Tidak	Tanggal terakhir logout.
10	Lastip	Varchar(15)	Tidak	Alamat IP terakhir login.
11	Nama	Varchar(60)	Tidak	Nama pengguna.

No	Nama *Field*	Tipe Data	*Null*	Keterangan
12	Email	Varchar(50)	Tidak	*Email* yang digunakan untuk mengirim *password* atau *mCode*.
13	Jk	Boolean	Tidak	Jenis kelamin pengguna
14	Alamat	Varchar(60)	Tidak	Alamat pengguna
15	Kota	Varchar(40)	Tidak	Kota tempat tinggal
16	Negara	Tinyint(3)	Tidak	Id negara
17	Reqkode	Varchar(14)	Ya	Daftar kode yang akan diminta ke pengguna untuk otentikasi tahap kedua.
18	a1	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label A1 (*hashed*).
19	a2	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label A2 (*hashed*).
20	a3	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label A3 (*hashed*).
21	a4	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label A4 (*hashed*).

No	Nama *Field*	Tipe Data	*Null*	Keterangan
22	a5	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label A5 (*hashed*).
23	a6	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label A6 (*hashed*).
24	b1	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label B1 (*hashed*).
25	b2	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label B2 (*hashed*).
26	b3	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label B3 (*hashed*).
27	b4	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label B4 (*hashed*).
28	b5	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label B5 (*hashed*).

No	Nama *Field*	Tipe Data	*Null*	Keterangan
29	b6	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label B6 (*hashed*).
30	c1	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label C1 (*hashed*).
31	c2	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label C2 (*hashed*).
32	c3	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label C3 (*hashed*).
33	c4	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label C4 (*hashed*).
34	c5	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label C5 (*hashed*).
35	c6	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label C6 (*hashed*).

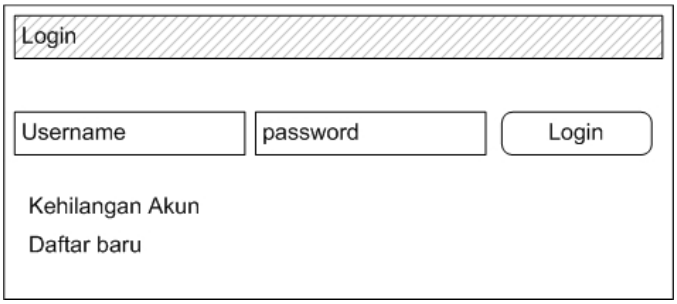
No	Nama *Field*	Tipe Data	*Null*	Keterangan
36	d1	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label D1 (*hashed*).
37	d2	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label D2 (*hashed*).
38	d3	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label D3 (*hashed*).
39	d4	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label D4 (*hashed*).
40	d5	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label D5 (*hashed*).
41	d6	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label D6 (*hashed*).
42	e1	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label E1 (*hashed*).

No	Nama *Field*	Tipe Data	*Null*	Keterangan
43	e2	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label E2 (*hashed*).
44	e3	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label E3 (*hashed*).
45	e4	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label E4 (*hashed*).
46	e5	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label E5 (*hashed*).
47	e6	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label E6 (*hashed*).
48	f1	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label F1 (*hashed*).
49	f2	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label F2 (*hashed*).

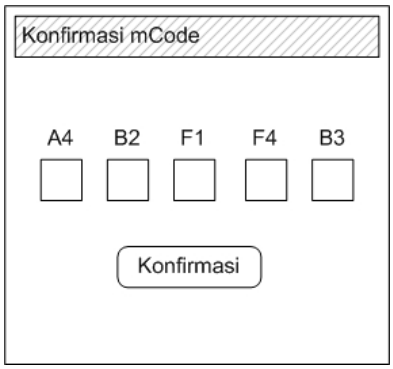
No	Nama *Field*	Tipe Data	*Null*	Keterangan
50	f3	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label F3 (*hashed*).
51	f4	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label F4 (*hashed*).
52	f5	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label F5 (*hashed*).
53	f6	Varchar(32)	Tidak	*Field* yang digunakan untuk menyimpan *mCode* dengan label F6 (*hashed*).

4.4.6. Perancangan *Interface*

Antarmuka *modAuth* terdiri dari beberapa antarmuka, antara lain *form Login*, *form* Daftar, *form* Ubah Data Pribadi dan beberapa *form* lainnya. Untuk *form Login* dan *form* Konfirmasi *mCode* dapat dilihat pada gambar 4.11 dan gambar 4.12. Sedangkan penjelasan detail dan rancangan *form* lainnya dapat dilihat pada lampiran B.



Gambar 4.11 Perancangan *Form Login modAuth*



Gambar 4.12 Perancangan *Form* Konfirmasi *mCode modAuth*

BAB V

IMPLEMENTASI DAN PENGUJIAN

Pada bab implementasi dan pengujian ini pembahasan akan dibagi menjadi dua bagian, yaitu pembahasan tentang implementasi dan pembahasan tentang pengujian.

5.1. Implementasi Modul

Implementasi *modAuth* meliputi pembuatan *database*, pengkodean *class* utama (*class modAuth*), pengkodean antarmuka, penggabungan semua kode yang dibuat, pengujian kode dan revisi atau perbaikan terhadap *bug* dan *error*.

modAuth diimplementasikan menjadi sebuah modul yang dapat digunakan oleh situs *web* sebagai menu otentikasinya. Pengkodean menggunakan pendekatan *Object Oriented Programming (OOP)* dengan bahasa pemrograman web PHP versi 5. *Database* yang digunakan adalah MySQL versi 5.

5.1.1. Lingkungan Implementasi

Lingkungan implementasi *modAuth* terbagi menjadi dua, yaitu lingkungan *hardware* dan lingkungan *software*. Berikut ini akan dipaparkan lingkungan implementasi *modAuth*.

1. Lingkungan *Hardware*

Lingkungan *hardware* (perangkat keras) untuk implementasi modAuth adalah sebagai berikut:

a. Prosesor : Intel® Pentium® 4 2,4 GHz

b. Memori : 512 MHz

c. *Harddisk* : SATA 160 GB (*master*) dan ATA 40 GB (*slave*)

2. Lingkungan *Software*

Lingkungan *software* (perangkat lunak) untuk implementasi *modAuth* adalah sebagai berikut:

a. *Web Server* : Apache Server 2.2.8 (*localhost*)

b. *PHP* : PHP 5.2.6

c. *Database Server* : MySQL 5.0.51b

d. *Web Browser* : Mozilla Forefox 3.0

5.1.2. Batasan Implementasi

Agar pelaksanaan implementasi tidak keluar dari fokus pembahasan, maka perlu dijelaskan batasan implementasi *modAuth* sebagai berikut:

1. Bahasa pemrograman web yang digunakan adalah *PHP* versi 5. Sedangkan *database MySQL* versi 5.
2. Prosedur Daftar dan *Reset mCode* belum sepenuhnya menjamin tidak dimanfaatkannya *username* dan *password* oleh orang lain. Akan diperlukan metode lain untuk prosedur Daftar dan *Reset mCode.* Karena proses kerja Daftar dan *Reset mCode* berbeda dengan proses kerja *Login*.
3. Contoh situs *web* dimana *modAuth* akan digunakan tidak termasuk ke dalam implementasi.

5.1.3. Hasil Implementasi

Pembahasan hasil implementasi ini dibagi menjadi dua bagian, yaitu dokumentasi terhadap *source code modAuth* dan tampilan antarmuka *modAuth*.

a. Dokumentasi *Source Code modAuth*

Hasil implementasi *modAuth* menghasilkan beberapa *file source code*. Penjelasan tentang *file source code* secara umum dapat dilihat pada tabel 5.1.

Tabel 5.1 Dokumentasi *File Source Code modAuth*

No	Nama file	Ukuran	Keterangan
1	modauthclass.php	29 kb	Merupakan *file* yang berisi *class modAuth*.
2	config-auth.php	1 kb	Merupakan *source code* untuk konfigurasi *modAuth*.
3	formDaftar.php	10 kb	Merupakan *source code* untuk *form* Daftar.
4	formLogin.php	8 kb	Merupakan *source code* untuk *form Login*, *form* Konfirmasi *mCode* dan *form Login* Sukses.
5	formKehilanganAkun.php	7 kb	Merupakan *source code* untuk *form Reset Password* dan *Reset mCode*.
6	formUbahmCode.php	3 kb	Merupakan *source code* untuk *form* Ubah *mCode.*
7	formUbahData.php	11 kb	Merupakan *source code* untuk *form* Ubah Data Pribadi.

No	Nama file	Ukuran	Keterangan
8	index.php	3 kb	Merupakan *source code* yang menjadi penghubung semua *file-file modAuth.*
9	loadcountry.php	1 kb	Merupakan *source code* untuk membuat *combo box* daftar negara.
10	md5engine.js	13 kb	Merupakan *script* yang berisi algoritma untuk konversi teks menjadi kode *MD5 Hash*.
11	style.css	2 kb	Merupakan *script* untuk mengatur tampilan atau *style modAuth*.

b. Tampilan Antarmuka *modAuth*

Tampilan antarmuka *modAuth* sebagian besar menggunakan *Cascade Style Sheet* (nama *file* adalah style.css).

Modul *modAuth* secara *default* akan memunculkan *form Login*, dari form ini pengguna akan dapat mengakses *form-form* lainnya. Berikut ini penjelasan tentang *form* yang ada pada *modAuth*.

Gambar 5.1 Hasil Implementasi *Interface Login* (*Form Default*)

Gambar 5.2 Hasil Implementasi *Interface* Konfirmasi *mCode*

Dari gambar 5.1 pengguna memiliki tiga pilihan aksi yang akan dilakukan, yaitu:

1. *Login* dengan memasukkan *username* dan *password* terlebih dahulu. Jika *username* dan *password* yang dimasukkan *valid*, maka pengguna akan dibawa ke *form* Konfirmasi *mCode*.
2. *Hyperlink* Bantu saya, merupakan akses ke *form Kehilangan Akun*. *Form* ini berfungsi untuk mendapatkan kembali *password* atau menciptakan *mCode* baru.
3. *Hyperlink* Daftar disini, merupakan akses ke form Daftar, dimana pengguna yang belum terdaftar dapat mendaftarkan sendiri akunnya.

Untuk penjelasan secara detail tentang implementasi *modAuth* dapat dilihat pada lampiran C.

5.2. Pengujian Modul

Untuk memudahkan dalam pengujian *modAuth*, maka pengujian *modAuth* akan dilakukan dalam dua bagian, yaitu pengujian fungsional dan pengujian non-fungsional.

5.2.1. Pengujian Fungsional

Pengujian ini bertujuan untuk menguji semua proses yang terjadi pada *modAuth* dan melihat kesesuaian dengan analisa yang dirancang. Pengujian akan dilakukan untuk setiap *use case* berdasarkan *sequence diagram*.

Hasil pengujian terhadap *use case Login* dapat dilihat pada tabel 5.2. Sedangkan hasil pengujian terhadap *use case* lain dapat dilihat pada lampiran E.

Tabel 5.2 Hasil Pengujian Terhadap *Use Case Login*

Keterangan :

✔ Implementasi sukses

No	Object (event)	Methods	Sequence Type	Input	Hasil yang diharapkan	Hasil pengujian	Kesimpulan
1	Tombol Login (onMouseOver/ onFocus)	onWillLogin()	Normal Flow/Exceptional Flow	Valid/ tidak valid	Password dienkrip menggunakan MD5 hash	Password dienkrip menggunakan MD5 hash	✔
2	Tombol Login (onSubmit)	validasiLogin()	Normal Flow/Exceptional Flow	Tidak valid	Muncul pesan kesalahan dalam pengisian form, lalu kembali ke form Login.	Muncul pesan kesalahan dalam pengisian form, lalu kembali ke form Login.	✔

No	Object (event)	Methods	Sequence Type	Input	Hasil yang diharapkan	Hasil pengujian	Kesimpulan
3	Tombol Login (onSubmit)	Login()	Normal Flow	valid	Muncul form Konfirmasi mCode.	Muncul form Konfirmasi mCode.	✔
4	Tombol Login (onSubmit)	Login()	Exceptional Flow(1)	Tidak valid	Muncul pesan kesalahan username dan password tidak ditemukan, lalu kembali ke form Login	Muncul pesan kesalahan username dan password tidak ditemukan, lalu kembali ke form Login	✔
5	Tombol Konfirmasi (onMouseOver/ onFocus)	onWillKonfirm()	Normal Flow/Exceptional Flow	Valid/ tidak valid	mCode dienkrip menggunakan MD5 Hash	mCode dienkrip menggunakan MD5 Hash	✔

No	Object (event)	Methods	Sequence Type	Input	Hasil yang diharapkan	Hasil pengujian	Kesimpulan
6	Tombol Konfirmasi (onSubmit)	validasi Konfirm()	Normal Flow/Exceptional Flow	Tidak valid	Muncul pesan kesalahan dalam pengisian form, lalu kembali ke form Konfirmasi mCode	Muncul pesan kesalahan dalam pengisian form, lalu kembali ke form Konfirmasi mCode	✔
7	Tombol Konfirmasi (onSubmit)	KonfirmasimCode()	Normal Flow	valid	Muncul form Login Sukses	Muncul form Login Sukses	✔
8	Tombol Konfirmasi (onSubmit)	KonfirmasimCode()	Exceptional Flow(2)	Tidak valid	Muncul pesan kesalahan mCode, lalu kembali ke form Login	Muncul pesan kesalahan mCode, lalu kembali ke form Login	✔

5.2.2. Pengujian Non-Fungsional

Pengujian ini bertujuan untuk melihat bahwa *modAuth* dapat menghasilkan kode acak setiap kali melakukan proses otentikasi, lalu mengirimkannya melalui jaringan. Sehingga aplikasi *keylogger* atau *sniffer* mencatat mCode yang berbeda-beda setiap kali melakukan proses otentikasi.

1. Aplikasi Penguji

Pada pengujian ini digunakan beberapa aplikasi *keylogger* dan *sniffer* untuk melihat paket data yang dikirim melalui jaringan. Aplikasi yang digunakan pada pengujian ini dapat dilihat pada tabel 5.3.

Tabel 5.3 Aplikasi Yang Digunakan Untuk Menguji *modAuth*

Informasi	**Aplikasi 1**	**Aplikasi 2**
Nama Aplikasi	Wireshark	OverSpy
Versi	0.99.6a	2.1
Tipe	*Network Analyzer (Sniffer)*	*Keylogger*
Lisensi	*Freeware*	*Commercial Demo*
Publisher	Gerald Combs	Virtuoza
Situs *web*	www.wireshark.org	-

2. Skenario Pengujian

Untuk mendapatkan hasil pengujian dengan tepat, maka diperlukan sebuah skenario pengujian, skenario pengujian *modAuth* adalah sebagai berikut:

a. Sebelum melakukan pengujian, *modAuth* dipasang ke sebuah *file* index.php yang diasumsikan sebagai sebuah situs *web*. Pemasangan tersebut bisa berupa hosting ke *server* atau pun *localhost* menggunakan dua komputer. Untuk pengujian ini menggunakan *domain* http://affandes.200u.com.

b. Setelah pemasangan, *modAuth* diatur dan diuji coba berjalan pada *server*, sehingga *modAuth* dapat berjalan dengan baik.

c. Dilakukan pemasangan (instalasi) terhadap aplikasi *keylogger* dan *sniffer* (lihat tabel 5.3).

d. Pengujian dilakukan hanya pada *form Login*, *form* Konfirmasi *mCode* dan *form Login* Sukses.

e. Sebelum dimulai pengujian, maka dipastikan semua aplikasi *keylogger* dan *sniffer* berjalan dengan baik.

f. Pengujian dilakukan sebanyak 10 kali proses otentikasi dengan *username* : **affan** dan *password* : **1234**. Sedangkan *mCode* dapat dilihat pada gambar 5.3.

g. Dalam memasukkan *username*, *password* atau *mCode* dilakukan secara normal atau tidak dilakukan pengacakan.

	A	B	C	D	E	F
1	c	7	c	w	U	V
2	E	X	n	J	5	5
3	f	T	e	H	m	A
4	F	X	q	6	i	o
5	q	o	q	X	a	4
6	Q	P	u	m	5	M

Gambar 5.3 Tabel *mCode* Pengguna

3. Hasil Pengujian

Setelah dilakukan pengujian maka diperoleh hasil pengujian seperti terlihat pada tabel 5.4. Sedangkan detail hasil pengujian dapat dilihat pada lampiran D.

Tabel 5.4 Hasil Pengujian Menggunakan Aplikasi Wireshark

Simbol ✔ berarti data yang diinputkan sebelumnya tidak bisa digunakan untuk *login* berikutnya.

Tahap Uji	**Data yang diperoleh menggunakan *Wireshark***				**Hasil**
	Username	***Password***	**Label *mCode***	**Nilai *mCode***	
1	affan	8de86fea 3ce5213f 84502fa4 e017a939	B4,F6, C3,D4, D1	15134111f51c78bd03ae2f5961cf72e8 ce66b199e9c590689096bc6f61e086c3 4b8a75037aff5e1c8f6c8994394365e4 b08b67698626594818e208e1c74f4fc1 3cc01b90f0b2c35947494441cb98f335	✔
2	affan	8de86fea 3ce5213f 84502fa4 e017a939	A5,B2, F6,C1, C5	c9d73a58dbcf266fec695f52e7383cef 15134111f51c78bd03ae2f5961cf72e8 ce66b199e9c590689096bc6f61e086c3 c570cf91cdc8d734dd27d03ea7bf1b74 c9d73a58dbcf266fec695f52e7383cef	✔
3	affan	8de86fea 3ce5213f 84502fa4 e017a939	D1,C1, F3,A4, E2	3cc01b90f0b2c35947494441cb98f335 c570cf91cdc8d734dd27d03ea7bf1b74 1e7d25f88fe31e67f62cf7b47a5829a7 9dd2aa6379dc31c53c12bcd8325bfeff bfa000e0d9d0405de0107d613631abcb	✔
4	affan	8de86fea 3ce5213f 84502fa4 e017a939	C2,A3, B4,F5, C2	feea05f2a90213c6384ed7b72c49674f 5c2ba8154f6233c57ac9be5b314307ca 15134111f51c78bd03ae2f5961cf72e8 4f49b5ed3709283589444dea37b94c1b feea05f2a90213c6384ed7b72c49674f	✔
5	affan	8de86fea 3ce5213f 84502fa4 e017a939	A4,C2, D4,E3, D2	9dd2aa6379dc31c53c12bcd8325bfeff feea05f2a90213c6384ed7b72c49674f b08b67698626594818e208e1c74f4fc1 941a69647fd03ce19d5398f6a668b851 1dcecd2f5a33b5b71b299bc5e4df50db	✔
6	affan	8de86fea 3ce5213f 84502fa4 e017a939	D2,E1, B4,B6, C2	1dcecd2f5a33b5b71b299bc5e4df50db 6806bf7f69bbaeaf8553ed19c3fb7c1e 15134111f51c78bd03ae2f5961cf72e8 150cb229623d899b6a952f8ec9f91682 feea05f2a90213c6384ed7b72c49674f	✔

Tahap Uji	Data yang diperoleh menggunakan *Wireshark*				Hasil
	Username	***Password***	**Label *mCode***	**Nilai *mCode***	
7	affan	8de86fea 3ce5213f 84502fa4 e017a939	E4,C1, A1,F1, A4	1ac9abc07671d59601eb58e114db29eb c570cf91cdc8d734dd27d03ea7bf1b74 c570cf91cdc8d734dd27d03ea7bf1b74 34347827d9191d8deff220d637398eb4 9dd2aa6379dc31c53c12bcd8325bfeff	✔
8	affan	8de86fea 3ce5213f 84502fa4 e017a939	F4,B5, F6,F5, D4	6c9bb9f08afe6ac2fd1e1c9fbd2a8a2d 6c9bb9f08afe6ac2fd1e1c9fbd2a8a2d ce66b199e9c590689096bc6f61e086c3 4f49b5ed3709283589444dea37b94c1b b08b67698626594818e208e1c74f4fc1	✔
9	affan	8de86fea 3ce5213f 84502fa4 e017a939	E2,C4, F3,C1, E6	bfa000e0d9d0405de0107d613631abcb c9d73a58dbcf266fec695f52e7383cef 1e7d25f88fe31e67f62cf7b47a5829a7 c570cf91cdc8d734dd27d03ea7bf1b74 bfa000e0d9d0405de0107d613631abcb	✔
10	affan	8de86fea 3ce5213f 84502fa4 e017a939	B3,B3, D5,D6, E4	c37d224dcff34e67f20f13cafaf88693 c37d224dcff34e67f20f13cafaf88693 15134111f51c78bd03ae2f5961cf72e8 941a69647fd03ce19d5398f6a668b851 1ac9abc07671d59601eb58e114db29eb	✔

Selanjutnya hasil pengujian menggunakan aplikasi OverSpy dapat dilihat pada tabel 5.5.

Tabel 5.5 Hasil Pengujian Menggunakan aplikasi OverSpy

Simbol ✔ berarti data yang diinputkan sebelumnya tidak bisa digunakan untuk *login* berikutnya.

Tahap Uji	Data yang diperoleh menggunakan *OverSpy*				Hasil
	Username	***Password***	**Label *mCode***	**Nilai *mCode***	
1	affan	1234	-	X,M,e,6,x	✔
2	affan	1234	-	q,X,M,c,q	✔
3	affan	1234	-	w,c,A,F,5	✔
4	affan	1234	-	n,f,X,4,n	✔
5	affan	1234	-	F,n,6,m,J	✔
6	affan	1234	-	J,U,X,P,n	✔
7	affan	1234	-	i,c,c,V,F	✔
8	affan	1234	-	o,o,M,4,6	✔
9	affan	1234	-	5,q,A,c,5	✔
10	affan	1234	-	T,T,X,m,i	✔

5.2.3. Kesimpulan Pengujian

Dari tabel-tabel hasil pengujian di atas dapat disimpulkan bahwa setiap data yang diinputkan pada *modAuth* akan berbeda-beda setiap kali melakukan otentikasi. Sehingga data tersebut tidak bisa dimanfaatkan untuk otentikasi selanjutnya.

Selain dari itu, dalam pengujian ini ditemukan suatu kelemahan. Dari tabel 5.4 terlihat jelas bahwa aplikasi *sniffer* mampu memperoleh semua data pengguna, mulai dari *username*, *password* (terenkripsi), label *mCode* sampai nilai *mCode* (terenkripsi). Jika dilakukan *monitoring* secara terus-menerus terhadap suatu akun, kemungkinan akan dapat terbaca semua tabel *mCode* akun tersebut. Misalnya pada otentikasi pertama pengguna diminta memasukkan *mCode* F3, D1, A6, A2, B1. Suatu saat salah satu atau bahkan semua *mCode* tersebut akan diminta kembali oleh *modAuth*.

BAB VI

PENUTUP

Penutup merupakan bagian akhir dari laporan tugas akhir yang berisi kesimpulan dan saran yang dapat dilihat melalui uraian berikut.

6.1. Kesimpulan

Kesimpulan dari hasil penelitian ini adalah:

1. Modul *Authentication* (*modAuth*) telah berhasil dirancang serta diimplementasikan dalam bentuk sebuah modul berbasis *web* yang dapat ditambahkan pada suatu situs *web*.
2. Berdasarkan pengujian terhadap konsep *lookup table* dapat disimpulkan bahwa *modAuth* telah mampu mencegah pengguna aplikasi *keylogger* dan aplikasi *sniffer* untuk memanfaatkan *username* dan *password*.
3. Di sisi lain terdapat suatu kelemahan yang memungkinkan *sniffer* merekam semua tabel *mCode* jika *sniffer* terus-menerus memantau suatu akun saat melakukan otentikasi ke *modAuth*.

6.2. Saran

Ada beberapa saran yang perlu diperhatikan dalam pengembangan *modAuth* selanjutnya, yaitu:

1. Penambahan metode pengamanan *username* dan *password* pada saat pendaftaran, ubah *password* dan ubah *mCode*. Ini diperlukan karena ada kemungkinan pencurian data *username* dan *password* dilakukan pada saat melakukan pendaftaran akun atau pada saat mengubah *password* dan *mCode*.
2. Implementasi menggunakan *Secure Socket Layer* (*SSL*) untuk meningkatkan enkripsi terhadap data yang dikirim melalui jaringan komputer.
3. Perubahan model dan metode penyimpanan *lookup table* ke *database* dengan metode yang lebih baik sehingga dihasilkan struktur *database* yang lebih efisien. Ini sangat berguna dalam merancang suatu sistem dengan kapasitas *database* yang besar. Selain itu, rancangan *database* yang baik juga akan menghasilkan performa yang baik pula.

DAFTAR PUSTAKA

Dhar, Summit. *"Sniffers – Basic and Detection"*. Reliance Infocomm. 2002.

Ivar Jacobson, Grady Booch, and Jim Rumbaugh. "*Unified Software Development Process"*. Addison-Wesley, 2002.

Klevinsky, T.J dan Scott Laliberte. *"Hack I.T. – Security Through Penetration Testing"*. Addison-Wesley. 2002.

M. Farid Azis. *"Belajar Sendiri Pemrograman PHP 4 Bagi Web Programmer"*. PT Elex Media Komputindo, Jakarta, 2001.

Philippe Kruchten. A Rational Development Process, "*CrossTalk"*, vol 9 (7), STSC, Hill AFB, UT, hal.11-16.

Philippe Kruchten. *Rational Unified Process—An Introduction*. Addison-Wesley, 1998.

Rational Team. *"Rational Unified Process : Best Practices for Software Development Teams"*. 2001.

Rivest, R. "*The MD5 Message-Digest Algorithm [RFC1321]"*. MIT. 1994.

Suhendar,A. *"Teknologi Pemograman Mobile Commerce"*. Penerbit Informatika, Bandung, 2002.

"*Keystroke logging*". Maret 2008 [*Online*] *Available http://en.wikipedia.org/wiki/Keystroke_logging*, diakses 20 Juli 2009.

"*Lookup table*". [*Online*] *Available http://en.wikipedia.org/wiki/lookup_table*, diakses 25 Oktober 2009.

"*MD5*". [*Online*] *Available http://en.wikipedia.org/wiki/MD5*, diakses 20 Juli 2009.